UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Departamento de Informática y Sistemas

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
DEPARTAMENTO DE INFORMÁTICA Y SISTEMAS

TESIS DOCTORAL

LOCALIZACIÓN DE CONTORNOS CON PRECISIÓN SUB-PIXEL EN IMÁGENES BIDIMENSIONALES Y TRIDIMENSIONALES

Agustín Trujillo Pino

Las Palmas de Gran Canaria, Octubre 2004

Anexo I

D. LUIS MAZORRA MANRIQUE DE LARA, SECRETARIO DEL DEPARTAMENTO DE INFORMÁTICA Y SISTEMAS DE LA UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

C E R T I F I C A,

Que el Consejo del Departamento en su sesión de fecha del 29 de Octubre de 2004 tomó el acuerdo de dar el consentimiento para su tramitación, a la tesis doctoral titulada "**LOCALIZACIÓN DE CONTORNOS CON PRECISIÓN SUB-PIXEL EN IMÁGENES BIDIMENSIONALES Y TRIDIMENSIONALES**" presentada por el doctorando D. Agustín Trujillo Pino y dirigida por el Doctor D. Luis Álvarez León y codirigida por los Doctores D. Julio Esclarín Monreal y D. Miguel Alemán Flores.

Y para que así conste, y a efectos de lo previsto en el Artº 73.2 del Reglamento de Estudios de Doctorado de esta Universidad, firmo la presente en

Las Palmas de Gran Canaria, a 29 de Octubre de 2004

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA · DEPARTAMENTO DE INFORMÁTICA Y SISTEMAS

Fdo. Luis Mazorra Manrique de Lara

Anexo II

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

Departamento: "Departamento de Informática y Sistemas"

Programa de Doctorado: "Percepción Artificial y Aplicaciones"

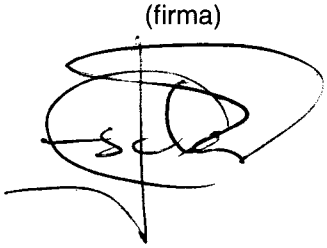
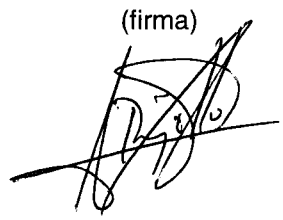
Título de la Tesis

LOCALIZACIÓN DE CONTORNOS CON PRECISIÓN SUB-PIXEL EN IMÁGENES BIDIMENSIONALES Y TRIDIMENSIONALES

Tesis Doctoral presentada por D. **Agustín Trujillo Pino**

Dirigida por el Dr. D. **Luis Álvarez León**

Codirigida por los Drs. D. **Julio Esclarín Monreal** y D. **Miguel Alemán Flores**

Director	Codirector	Codirector	Doctorando
(firma)	(firma)	(firma)	(firma)
			
Luis Álvarez León	Julio Esclarín Monreal	Miguel Alemán Flores	Agustín Trujillo Pino

TESIS DOCTORAL

Autor:	**Agustín Trujillo Pino**
Director:	**Luis Álvarez León**
Codirectores:	**Julio Esclarín Monreal** **Miguel Alemán Flores**
Título:	**Localización de Contornos con Precisión Sub-Pixel en Imágenes Bidimensionales y Tridimensionales**
Universidad:	**Universidad de Las Palmas de Gran Canaria**
Departamento:	**Departamento de Informática y Sistemas**
Programa:	**Percepción Artificial y Aplicaciones**
Grado:	**Doctor en Informática**
Fecha:	**18 de Octubre de 2004**

Tesis doctoral presentada por **D. Agustín Trujillo Pino**, y dirigida por el profesor **D. Luis Álvarez León** y codirigida por los profesores **D. Julio Esclarín Monreal** y **D. Miguel Alemán Flores**, para la obtención del grado de **Doctor en Informática**.

Índice General

A Claudia, mi mujer,
que me ha dado tres hijos maravillosos,
que cuida de mí y de todos ellos
y hace que parezca tarea fácil.

Ella aguanta mis defectos,
me anima en mis momentos bajos,
me apoya en lo que hago,
y me escribe las dedicatorias.

Agradecimientos

La presente tesis representa el resultado de todo el trabajo de investigación que he realizado dentro del Departamento de Informática y Sistemas en los últimos años. Y como todas las grandes metas que uno se plantea en el transcurso de la vida, se ha hecho de rogar, y bastante. Pero al final se ha conseguido.

Puede decirse que todo comenzó en Octubre de 1991, fecha en la que entré a la Universidad como profesor asociado, coincidiendo con la creación del grupo de investigación Análisis Matemático de Imágenes liderado por el doctor **Luis Álvarez**. Aunque ese año me centré más en la docencia, ya comencé a dar mis primeros pasos en la investigación. A Luis le considero por tanto mi maestro en el mundo de la investigación, y por ello y por albergarme dentro de su grupo le estaré eternamente agradecido. El haber podido concluir esta tesis es casi más mérito suyo que mío.

Dentro de dicho grupo, mi actividad inicial durante los años 92, 93 y 94 fue participar de lleno en el desarrollo del software XMegaWave (XMW) para el tratamiento de imágenes, con varias estancias en la Universidad de Las Islas Baleares bajo la dirección de Josep Blat, y en la Universidad de París Dauphine colaborando con el investigador Jacques Froment. Y simultáneamente al desarrollo del software, realicé los cursos de doctorado y comencé a escribir los primeros artículos. Todo este trabajo, incluyendo las estancias fuera de la isla, lo realicé junto a mi compañera **Esther González**, a la cual quiero agradecerle desde aquí su paciencia por aguantarme todo este tiempo, y porque sé que a veces trabajar conmigo no es tarea fácil.

A continuación hubieron cuatro acontecimientos que ralentizaron mi actividad investigadora. En el 95 y tras un ultimatum de mi novia pasé por la vicaría, y ya se sabe el tiempo que quita semejante acontecimiento. En el 96 el Estado Español me mandó a presidio para realizar mi prestación social, impartiendo clases a los reclusos: fue toda una experiencia. En el 97 preparé y aprobé la oposición como profesor Titular de Escuela Universitaria.

El último acontecimiento, y el que más tiempo me llevó, fue durante los años 98 y 99 en los que ejercí el cargo de Jefe de Estudios en la Escuela de Informática, coincidiendo con la puesta en marcha de las nuevas titulaciones de Ingeniería en Informática. De esta última época lo único que recuerdo es hacer horarios y pelearme con el resto de profesores.

Aún así, en esos 5 años pude también continuar con el desarrollo de nuevas funciones para el XMW, y realizar algunos trabajos aceptados en congresos por esas fechas. Ya en ese tiempo me había centrado en la problemática de la detección de bordes en una imagen. Entre otras cosas, el sombrero de Lenna (lady.rim) ya no tenía secretos para mí, y todos sus pixels me los conocía al dedillo. De esta época tengo mucho que agradecerle a **Julio Esclarín**, por forzarme a continuar con la investigación y enseñarme cómo moverme en el mundo de los congresos y de los proyectos de investigación.

En los años 2000 y 2001 tuve la suerte de convivir "laboralmente" con el investigador francés **Karl Kris-**

sian, que realizó una estancia de un año en nuestro laboratorio, prorrogada varios meses después de que éste conociera los carnavales y las playas de Pozo Izquierdo. Con él participé en la creación de un nuevo software, denominado AMILab, combinando el código que cada uno de nosotros había desarrollado en los últimos años. Su software estaba orientado a imágenes 3D, y fue él quien me animó a trabajar en este tipo de imágenes.

Durante esa época fue cuando comenzó a tomar forma la investigación que ahora se presenta en este trabajo, y desde entonces hasta ahora dicha tarea ha acaparado prácticamente toda mi jornada laboral, si exceptuamos las horas de docencia.

En estos últimos años he recibido ayuda de muchos compañeros, pero en especial de **Miguel Alemán**. Si alguien se ha leído y releído las 383 páginas de esta obra, y corregido cada coma y cada tilde, ése ha sido él. A Miguel le agradezco enormemente no sólo su labor de corrección, sino también su colaboración y ayuda en todo momento durante casi todas las fases de la investigación por las que he ido pasando, así como unas cuantas sesiones de psiquiatría en mis momentos más bajos.

Por último, mi eterno agradecimiento a mis padres y "abuelos", sin los cuales no hubiera llegado hasta aquí, y a mi mujer Claudia y a mis tres retoños, que me enseñaron que no todo en la vida son pixels.

OCTUBRE 2004

Introducción

El tratamiento de imágenes digitales es una línea de investigación enmarcada dentro del campo de la visión artificial por ordenador. La detección de bordes en una imagen es uno de sus objetivos más importantes, como paso previo para la medición o el reconocimiento de información de más alto nivel. La presente tesis consiste en un estudio en profundidad de la naturaleza de este problema, a partir del cual deduciremos las distintas técnicas que propondremos para resolverlo de forma precisa.

El sistema visual humano interpreta y reconoce los objetos principalmente a través de su silueta, y de los cambios bruscos de contraste o de color en su interior. Este salto brusco en los valores de intensidad de la imagen es lo que se denomina **borde**, e indica las fronteras o líneas de separación entre los distintos objetos presentes en la imagen. Por lo tanto, detectar bordes con la mayor precisión posible es uno de los procesos fundamentales de la visión artificial.

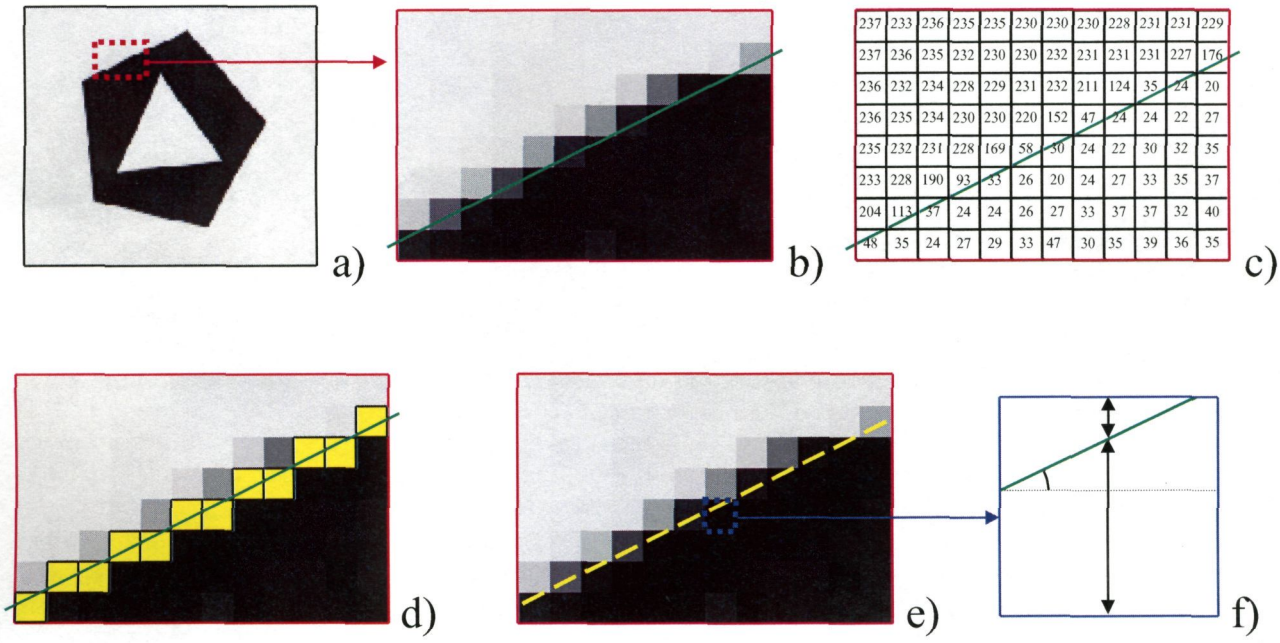
El campo de aplicación para este tipo de técnicas es elevadísimo. En general, puede aplicarse en cualquier sistema o proceso de cualquier índole, en donde exista como dato de entrada una o varias imágenes que puedan pasarse a formato digital, y sobre la cual se necesiten medir ciertos parámetros, interpretar visualmente la escena representada en ella, o reconocer algunos patrones previamente conocidos: detectar piezas concretas sobre una cinta transportadora dentro de un entorno industrial, reconocer caras en una cámara de vídeo que dé acceso de entrada a algún recinto, identificar bancos de peces en una foto aérea, reconocer huellas dactilares, identificar productos defectuosos con una cámara en una planta industrial, etc.

En el presente capítulo vamos a describir brevemente el problema que se quiere resolver en esta tesis, y daremos una descripción general de cómo se pretende abordarlo. Posteriormente daremos una visión general del estado del arte dentro de este campo, y comentaremos las aportaciones originales principales de este trabajo. También haremos breve mención del software que ha sido desarrollado durante el tiempo que ha durado esta investigación, y sin el cual no hubiera sido posible la implementación y visualización de los resultados obtenidos. Finalmente detallaremos cómo se encuentra estructurado el resto de capítulos, y comentaremos algunas consideraciones que el autor desea hacer constar.

0.1 Descripción del problema a resolver

Una imagen digital almacenada en la memoria de un ordenador no es más que una matriz rectangular de valores, donde cada uno representa la intensidad de luz adquirida por cada uno de los miles de sensores presentes en el dispositivo de adquisición, como puede verse en la parte superior de la figura 1. Un borde por lo tanto puede considerarse como una línea virtual en la imagen (dibujada en color verde en la figura) que delimita dos zonas de intensidad diferente a ambos lados.

Normalmente la mayoría de detectores de borde en la literatura indican sólamente aquellos pixels considerados pertenecientes al borde, como en la figura 1d. Para ello parten de la información numérica de una cierta vecindad alrededor del pixel que se quiere estudiar, y deducen si ese pixel puede ser etiquetado como "perteneciente al borde". El resultado, debido a la naturaleza discreta de los pixels, tiene un efecto de aliasing



237	233	236	235	235	230	230	230	228	231	231	229
237	236	235	232	230	230	232	231	231	231	227	176
236	232	234	228	229	231	232	211	124	35	24	20
236	235	234	230	230	220	152	47	24	24	22	27
235	232	231	228	169	58	30	24	22	30	32	35
233	228	190	93	33	26	20	24	27	33	35	37
204	113	37	24	24	26	27	33	37	37	32	40
48	35	24	27	29	33	47	30	35	39	36	35

Figura 1: a) fotografía de un objeto oscuro sobre fondo claro; b) detalle ampliado de un borde; c) valores de los pixels en la imagen; d) detección de bordes a nivel pixel; e) detección de bordes a nivel subpixel; f) detalle de la detección a nivel subpixel

no deseado. Por otro lado, la información sobre las características de dicho borde, como puede ser la orientación, o la diferencia de intensidad presente a ambos lados de la imagen, dista mucho de ser exacta.

El problema que aquí queremos plantear es más complejo, ya que consiste en realizar la detección a nivel sub-pixel, como se muestra en la figura 1e. De esta forma, para poder lograr una precisión óptima, la estimación de la posición y orientación de los bordes se hará en el interior del pixel, como muestra la figura 1f.

La pregunta fundamental que se plantea y que da pie a todo este trabajo es la siguiente: dado un pixel perteneciente a un borde, del que sólo conocemos la información numérica de la intensidad de los pixels en una cierta vecindad, ¿será posible encontrar un esquema que detecte con total precisión la **localización exacta de dicho borde**, así como otra información característica como puede ser su orientación y su curvatura? Entiéndase que hablaremos de precisión total sólamente para casos ideales.

Para tratar de encontrar respuesta a esta pregunta, podemos hacer una abstracción y centrarnos simplemente en el problema numérico desde un punto de vista puramente matemático, partiendo de un modelo donde el valor concreto dentro de cada pixel está relacionado con el área de cada zona de la escena que se quiere fotografiar, y que resulta proyectado dentro del sensor asociado a dicho pixel. De esta forma, aplicando conceptos geométricos básicos podemos abordar el problema.

Posteriormente, la misma pregunta puede plantearse para imágenes 3D, en donde la información de la escena viene dada por una matriz tridimensional de valores, y en las cuales los bordes ya no son líneas sino superficies. De la búsqueda de la respuesta a esta pregunta es de lo que tratan los próximos diez capítulos.

0.2 Descripción general del trabajo realizado

El objetivo principal de esta tesis doctoral consiste por tanto en la detección precisa de los contornos en imágenes 2D y 3D, haciendo especial hincapié en las imágenes médicas, concretamente en imágenes angiográficas, como la de la figura 2. Es importante mencionar en este punto que, aunque nos centremos en este tipo de imágenes médicas, todos los conceptos y técnicas que se van a exponer en esta tesis son generalistas, en tanto en cuanto pueden aplicarse a cualquier tipo de imagen 2D ó 3D. De hecho, en algunos momentos también usaremos imágenes de otros ámbitos.

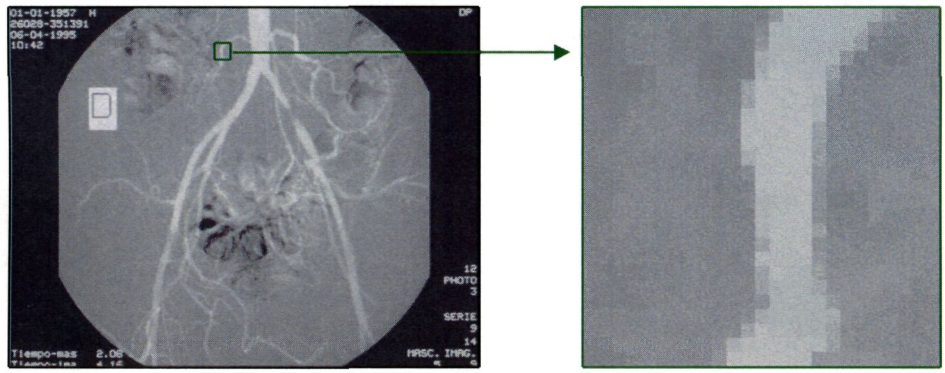


Figura 2: Image angiográfica y detalle ampliado

Una de las razones por las que hemos elegido las imágenes médicas radica en la enorme utilidad de la detección en este tipo de imágenes. No en vano el área de imagen médica es uno de los campos que más interés ha despertado en las últimas décadas, ya que continuamente se están usando todo tipo de imágenes (radiografías, scanners, tomografías, ecografías, etc.) para su análisis e interpretación por parte del personal médico. Por tanto, uno de los objetivos principales en este campo consiste en aportar la mayor información posible al médico para ayudar a éste en su diagnóstico. Otro objetivo también importante es permitir automatizar la medición de ciertas características en grandes conjuntos de imágenes, con vistas a que el personal médico pueda realizar estudios estadísticos de pacientes que, de otra forma, serían mucho más costosos de realizar. Por ejemplo, estimar medidas como el área de un tumor, o como el diámetro de una vena, son cálculos típicos que requieren de una detección de bordes precisa.

Dentro de las imágenes médicas, vamos a centrarnos en imágenes angiográficas o angiografías. Este tipo de imagen no es más que una fotografía de una zona de vasos sanguíneos usando rayos X, como la que se ve en la figura 2. La detección precisa de los contornos o bordes de los vasos en este tipo de imágenes es un paso previo fundamental para poder estimar medidas concretas sobre la vasculatura, como por ejemplo el grosor o la curvatura de los vasos en cada pixel, lo cual permitiría dar al médico un diagnóstico más preciso. Incluso podrían tomarse una serie de medidas cuantitativas del paciente, y asociarlo a su expediente de forma semi-automática.

Ya comentamos que aunque la mayoría de los detectores de borde trabajan a nivel pixel, como en la parte izquierda de la figura 3, nosotros vamos a proponer una detección a nivel sub-pixel, como en la parte derecha de la figura 3. De esta forma, las medidas dadas al médico serán más realistas. Por ejemplo, si se quisiera obtener una función que indicase la variación del grosor del vaso a medida que se avanza por él, con una detección a nivel pixel obtendríamos una función discreta a saltos (con valores enteros), mientras que con una detección a nivel subpixel la función tendría un perfil mucho más continuo y preciso.

Otra de las razones por la que se han elegido imágenes médicas, y concretamente angiografías, es porque este tipo de imágenes suele abarcar diferentes casos y configuraciones de bordes. Por ejemplo, un contorno tan claro

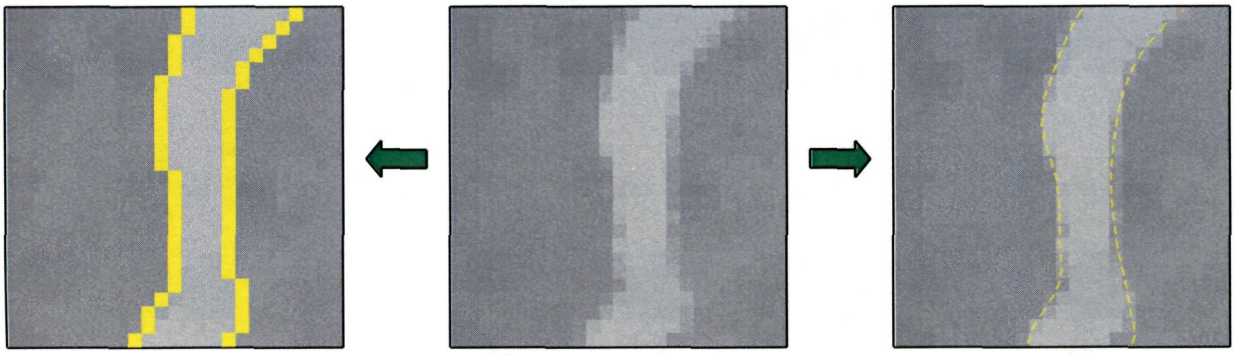


Figura 3: Contornos detectados en una angiografía a nivel pixel (imagen izquierda) y a nivel subpixel (imagen derecha)

como el de la figura 1b, que se encuentra aislado de otros contornos, donde apenas hay curvatura, y en donde hay un salto de intensidad bastante alto, no va a ser una situación muy común. Lo habitual en una angiografía será encontrar venas estrechas (contornos muy pegados entre sí), bifurcaciones, zonas con alta curvatura, etc. Además, en algunos casos, la calidad de la imagen no siempre va a ser perfecta, debido al ruido presente en el proceso de adquisición. A lo largo del trabajo iremos abordando todas estas situaciones y planteando nuevas modificaciones a los métodos propuestos para seguir manteniendo la precisión en la detección.

Es importante reseñar también que los parámetros que se quieren obtener son individuales para cada pixel que pertenece al borde. Nuestro objetivo se centra en **buscar la precisión en la medida de las características del borde en el interior de cada pixel** (posición, orientación, curvatura) como dato de entrada para otras aplicaciones de más alto nivel, como pueden ser métodos que realicen procesos de segmentación, en los que cada pixel de la imagen es identificado como perteneciente al interior o al exterior del vaso, o métodos detectores que tratan de obtener una línea cerrada representando la silueta completa de los objetos, como los trabajos que se citan en la sección 0.3.3. Todo este tipo de métodos podrían tomar como entrada los resultados de nuestro trabajo.

Todo el estudio realizado en esta tesis puede agruparse en tres grandes áreas que describimos a continuación.

0.2.1 Detección de bordes en imágenes 2D

Como ya se mencionó antes, la detección de bordes es uno de los procesos a bajo nivel más importantes dentro del tratamiento de imágenes, como etapa previa para la detección de parámetros de más alto nivel. Por lo tanto, es un área que lleva siendo estudiada bastantes años, y aún continúa desarrollándose. Comenzando por los trabajos fundamentales de Marr-Hildreth [MAR82] y Canny [CAN86], hasta trabajos más recientes, como los comentados en la sección 0.3.1. Ya indicamos que la mayoría de estos trabajos de detección de bordes están enfocados a nivel pixel. Sin embargo, cuando se requiere una precisión más fina, habría que tratar de encontrar la posición exacta del contorno en el interior de cada pixel.

Por tanto, el objetivo inicial de esta tesis consiste en tratar de localizar con precisión la posición, orientación y curvatura del contorno en el interior de cada pixel por el que dicho contorno atraviesa. Para ello, se van a proponer varios algoritmos novedosos que, con un coste computacional no demasiado alto, permitan obtener medidas de error muy pequeñas.

La idea de partida de los algoritmos desarrollados ha sido considerar un modelo de borde donde se asume la existencia de discontinuidad en los valores de la imagen, y un modelo de adquisición donde se considera el efecto de área de cada valor de intensidad a ambos lados del contorno en el interior del pixel. Con dicha idea, y

usando principalmente conceptos geométricos básicos, se ha desarrollado primero un método de grado 1, en el que los bordes son aproximados localmente en la vecindad de un pixel como segmentos rectos, y finalmente de grado 2, en el que la aproximación elegida son segmentos parábolicos (ver figura 4).

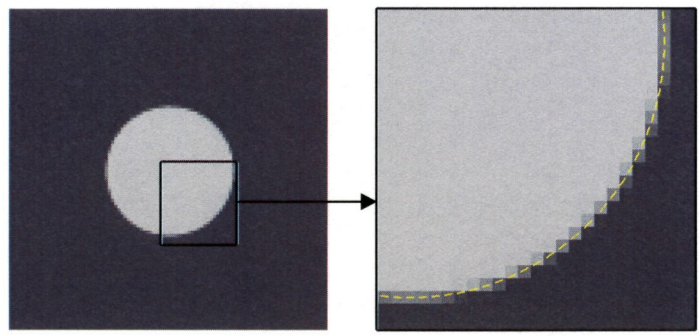


Figura 4: Contornos detectados a nivel subpixel sobre la imagen de un círculo

0.2.2 Detección de bordes en imágenes 3D

En los últimos años, con los nuevos dispositivos presentes en la mayoría de los hospitales (scanners 3D, resonancia magnética), ya es posible obtener imágenes tridimensionales, donde en lugar de tener una matriz bidimensional de pixels (imagen tradicional 2D) tenemos una matriz tridimensional de voxels (imagen 3D) (ver figura 5). La generalización de los trabajos hechos para dos dimensiones para adaptarlos a las 3 dimensiones implica mayor complejidad, como lo demuestra la comparativa hecha por Nikolaidis [NIK01].

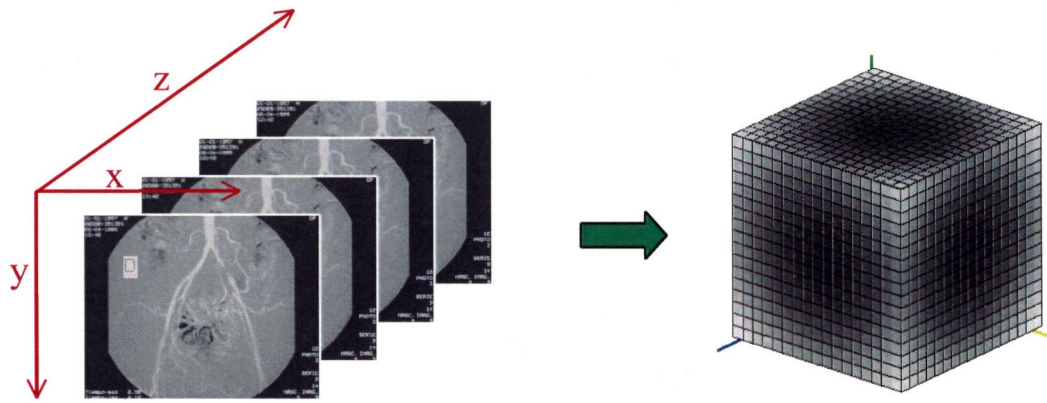


Figura 5: Una imagen 3D puede interpretarse como una lista de imágenes 2D seccionales

Al igual que hicimos en dos dimensiones, el principal objetivo de esta parte es tratar de localizar con precisión la posición, orientación y curvaturas del contorno en el interior de cada voxel por el que dicho contorno atraviesa, y para ello se propondrán también varios algoritmos. La idea de partida es la extrapolación de los modelos de borde y adquisición utilizados en el caso 2D. Hay que tener en cuenta que la diferencia fundamental es que, si antes un borde venía dado por la línea que separaba valores diferentes de intensidad en la imagen, en una

imagen 3D el borde viene dado por una superficie. Por tanto, el modelo de adquisición considera el efecto de volumen de cada valor de intensidad a ambos lados del contorno en el interior del voxel, también conocido por otros autores como PVE (partial volume effect) [SUE02]. Con dicha idea, y usando de nuevo conceptos geométricos tridimensionales básicos, se ha desarrollado primero un método de grado 1, en el que los bordes son aproximados localmente en la vecindad de un voxel como planos, y finalmente de grado 2, en el que la aproximación elegida son paraboloides (ver figura 6).

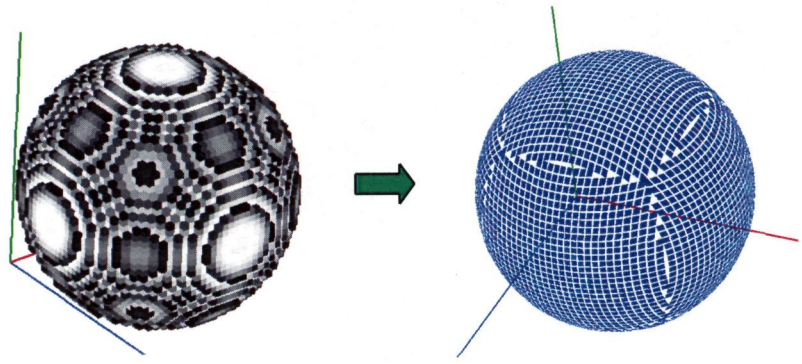


Figura 6: Contornos detectados a nivel subvoxel sobre la imagen 3D de una esfera

0.2.3 Eliminación de ruido

Es un hecho comprobado que el proceso de adquisición de la imagen introduce un cierto nivel de ruido debido a los procesos ópticos y magnéticos durante la adquisición. Por lo tanto, es preciso procesar la imagen previamente para tratar de eliminar en lo posible dicho ruido. Para poder aplicar los algoritmos propuestos en imágenes reales, se ha tenido que desarrollar también un esquema eficiente de eliminación de ruido para mejorar la calidad de las imágenes, antes de aplicar las técnicas de cálculo de características mencionadas antes.

Para ello, nos hemos basado principalmente en aplicar inicialmente un suavizado gaussiano a la imagen, y posteriormente analizar cómo es el comportamiento de una imagen con borde previamente suavizada, para tratar de detectar de forma exacta el borde original en la nueva imagen. La idea de partida es la siguiente: si a nuestros métodos iniciales le pasamos una imagen con ruido, el resultado será bastante caótico, debido a los valores erróneos en la intensidad de los pixels. Sin embargo, al desarrollar un nuevo método que acepte como entrada una imagen previamente suavizada, y que obtenga también con total precisión las características del borde original, significará que dispondremos de un método más robusto que dé resultados fiables sobre imágenes de cualquier calidad. Esto es debido a que el parecido entre el suavizado de una imagen con ruido y sin ruido, es mucho mayor que el que existe entre ambas imágenes antes de suavizar, con lo cual las estimaciones obtenidas para los parámetros del borde son más precisas.

Este nuevo esquema de eliminación de ruido puede aplicarse de forma recursiva, generando en cada iteración una nueva imagen sintética a partir de la información obtenida por el detector de bordes propuesto. Así, somos capaces de poder detectar bordes con gran precisión incluso cuando la magnitud del ruido es elevada (ver figura 7).

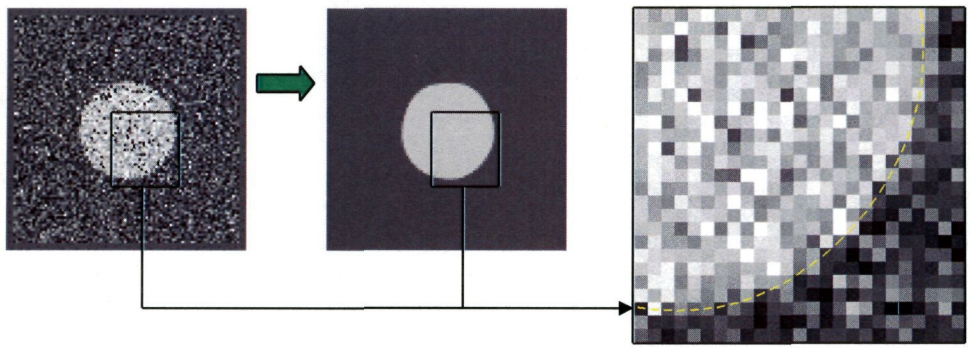


Figura 7: De izquierda a derecha: imagen con ruido, imagen restaurada, contornos detectados a nivel subpixel.

0.3 Estado del arte en la detección de bordes

A continuación expondremos las principales técnicas utilizadas en la literatura para la detección de bordes en imágenes 2D y 3D, y posteriormente comentaremos también las que están orientadas a imágenes angiográficas.

0.3.1 Detección de bordes en imágenes 2D

La mayoría de las técnicas de detección de bordes a nivel sub-pixel suelen seguir un proceso de dos etapas. En primer lugar realizan una detección a nivel pixel, utilizando esquemas clásicos de cálculo de gradiente como el de Canny [CAN86], Sobel [SOB70] o Shen-Castan [SHE92], para localizar máximos locales y detectar los pixels borde. En segundo lugar, se refina la localización para obtener una resolución a nivel subpixel. Estos algoritmos suelen aplicarse de forma local en la vecindad de los pixels borde detectados previamente, para reducir el coste computacional.

Para ver un análisis en profundidad de las técnicas existentes, así como una comparativa entre dichas técnicas, puede consultarse la bibliografía ([VEN03, WAL94, WES90]). Realizando una clasificación de los distintos algoritmos, vemos que pueden agruparse en tres categorías diferentes: las basadas en técnicas de reconstrucción, las que utilizan técnicas de interpolación, y las basadas en el cálculo de momentos.

Técnicas de reconstrucción

El objetivo de estas técnicas consiste en reconstruir la función de gradiente continua a partir de los valores discretos obtenidos al final de la detección a nivel pixel. Para lograr este objetivo, varios autores han usado diferentes funciones de reconstrucción. Una de las más obvias es la función *sinc*, la cual garantiza una reconstrucción óptima, si se asume el criterio de muestreo de Nyquist en la conversión analógica / digital. El principal problema de esta técnica es que las funciónes *sinc* decaen muy lentamente, lo que llevaría a tener que usar varias de ellas para poder aproximar la función gradiente a un nivel de precisión aceptable. Esto causaría un coste computacional inaceptable para la mayoría de las aplicaciones.

Sengupta y otros [SEN02] usan una aproximación más simple para resolver este problema. En lugar de reconstruir mediante la proyección en funciones *sinc*, hacen uso de la dualidad de los dominios espacial y frecuencial para realizar una interpolación en el dominio de la frecuencia. Asumiendo que la señal permanece limitada a la mitad de la frecuencia de muestreo, se puede incrementar la resolución rellenando con ceros las altas frecuencias, y entonces realizar la inversión al dominio espacial. Esto produce el efecto de interpolación en el dominio espacial usando la función *sinc* sin el coste computacional de ejecutar dicha operación. Además de la eficiencia obtenida, otra ventaja es que las altas frecuencias se suelen preservar mejor, con lo cual la precisión

suele ser más alta que usando interpolación espacial con un criterio de reconstrucción como puede ser el de Shannon. Finalmente, la posición real del borde se determina a partir de la imagen inversa sobremuestreada, lo cual aumenta automáticamente la resolución espacial a un nivel subpixel.

Una ligera modificación de la técnica anterior consiste en usar funciones base gaussianas [SEI88]. La idea es que en lugar de reconstruir la función gradiente original, se reconstruye una versión suavizada de la función utilizando núcleos gaussianos de anchuras convenientes, de forma que sólo haya que usar unas pocas gaussianas, y además se pueda asegurar que el suavizado no sea excesivo. Los autores usan una desviación de $\pi/4$ y muestran que con sólo cinco términos es suficiente para realizar la reconstrucción con una alta precisión.

Otra variación diferentes es el uso de splines como funciones de reconstrucción [TRU01b]. Los autores intentan estimar el gradiente espacial de manera robusta, aplicando primero el filtro de gradiente de Shen-Castan para obtener su perfil discreto, y posteriormente reconstruir el perfil continuo usando B-splines. El orden de los B-splines utilizados determinará la resolución conseguida, a costa de incrementar el coste computacional.

Una idea casi idéntica es propuesta por Li y otros en [LI88]. Los autores usan un esquema de detección a nivel pixel para obtener la posición, buscando la correspondencia con una plantilla de borde ideal, y entonces refinan este dato usando polinomios de Chebyshev como funciones de reconstrucción.

Sebastián-Zúñiga y otros [SEB97] enfocan el problema desde un punto de vista diferente. Los autores sostienen que existen dos fuentes de error principales e igualmente importantes en el proceso de determinación de los bordes. La primera es el difuminado que ocurre debido al proceso de adquisición, y la segunda es el proceso de muestreo y digitalización. El objetivo es por tanto eliminar o reducir el error total, recuperando completamente la señal ideal de entrada. Para ello usan un modelo de integración local para el sistema de adquisición, realizando la formulación en el dominio de la transformada de Laplace, lo que permite una manipulación más sencilla de las señales implicadas.

Técnicas de interpolación

En esta clase de técnicas, donde están incluidas las aproximaciones más extendidas, se trata de interpolar la función gradiente en el interior de una vecindad del máximo local usando un modelo conveniente. La principal diferencia entre éstas y las técnicas de reconstrucción es que los parámetros de la función de interpolación sólo son válidos en la vecindad del máximo local, y no para el perfil del gradiente completo como se asumía en las otras. La ventaja es que, al estar trabajando con una pequeña región de pixels, podemos encontrar una correspondencia mucho más exacta para los valores discretos del gradiente.

La función de interpolación más utilizada es la polinómica. El factor importante consiste en incluir solamente los puntos discretos relevantes para determinar los coeficientes del polinomio. Usar demasiados puntos suele producir resultados falseados, ya que un polinomio general puede modelar casi cualquier función, y la información resultante puede no ser muy útil. La aproximación más utilizada suele ser usar una interpolación cuadrática usando tres puntos alrededor del máximo local [LI88, SEI88].

La ventaja de estas técnicas es que, una vez hemos calculado los valores del polinomio, podemos encontrar la posición exacta del borde de forma analítica, lo que reduce el coste computacional total.

Técnicas basadas en el cálculo de momentos

Las técnicas de esta categoría usan la información derivada del gradiente de la imagen para calcular medidas estadísticas tales como los momentos del nivel de gris, momentos espaciales, valores del centroide, valores de energía local, valores esperados, etc. En el caso del centroide y de los valores esperados, la posición del borde se obtiene directamente a partir de los valores calculados por la aplicación de un valor umbral convenientemente fijado. En casi todos los casos, se trata de hacer corresponder los valores obtenidos con un conjunto de medidas derivadas a partir de un modelo de borde prefijado, y mientras, dicho modelo se va refinando hasta obtener una correspondencia óptima. La posición exacta del borde viene entonces dada por el modelo que mejor se ajuste a los datos de la imagen. Por lo tanto, este tipo de métodos requieren un conocimiento a priori de los datos para

poder ser de utilidad. Sin embargo, todos estos métodos tienen la ventaja de ser relativamente robustos, ya que se basan en operaciones de integración (sumas).

Un algoritmo de detección simple pero muy efectivo es el de cálculo del centroide [CLA93]. Primero se usa un filtro de gradiente para obtener el perfil del gradiente discreto a nivel pixel, y después se utiliza el centroide de gravedad de una vecindad alrededor de la posición del máximo local para determinar la localización del borde. Este método y sus variantes son de los más referenciados en la literatura.

Otro método muy popular también es el uso de los momentos de nivel de gris. Se define el momento de nivel de gris de orden n a la suma de las potencias de orden n de los valores de intensidad de la imagen. Este método fue originalmente utilizado por Tabatabai y Mitchell [TAB84], y posteriormente estudiado en más profundidad por otros investigadores [COS92]. El algoritmo original calcula los 3 primeros momentos de nivel de gris de una fila de la imagen que contenga el borde discreto, y luego busca el borde ideal que se corresponde con esos momentos. El método es invariante a cambios aditivos y multiplicativos de los valores de la imagen.

Una variación a la técnica de momentos es el uso de los valores esperados [GON92]. En primer lugar, se aislan las regiones que contienen bordes aplicando valores umbrales para la magnitud de la primera derivada de la señal. Estos umbrales han de ser los suficientemente bajos para asegurar que todos los posibles bordes son tenidos en cuenta. Además, se asume que los valores de gradiente en cada región se corresponden con una cierta distribución de probabilidad. Entonces, los valores esperados de la distribución calculados a partir de los valores del gradiente indicarán la posición exacta del borde, asumiendo que buscamos un borde ideal 1D convolucionado con una gaussiana sin ruido alguno.

Otra aproximación ligeramente diferente consiste en el cálculo de medidas de energía local [KIS94]. La principal idea es que se puede usar la respuesta de la señal del borde a un conjunto de filtros de energía local previamente elegidos para caracterizar un modelo de borde. Entonces, dicho modelo se ajusta a la señal realizando una minimización de la energía. Los autores usan un conjunto de filtros de cuadratura de Cauchy para realizar la caracterización. Cada tipo de perfil (rampa, escalón, valor constante) tiene una única respuesta cuando se convolucionan con estos filtros. Los autores definen 8 tipos de borde, y clasifican la señal de entrada como perteneciente a uno u otro de estos tipos según su respuesta a los filtros. Una vez se ha identificado el tipo, se ajusta el modelo de borde correspondiente a ese tipo con la señal de entrada utilizando una optimización por mínimos cuadrados de la energía local. Este proceso se realiza de forma iterativa donde, en cada iteración, la posición del borde en el modelo se va modificando hasta lograr un ajuste óptimo.

0.3.2 Detección de bordes en imágenes 3D

En una imagen tridimensional, el concepto de borde es diferente: ahora se trata de superficies en lugar de líneas. Para detectar dicha superficie con precisión, el proceso es similar al realizado en dos dimensiones: detectar en primer lugar los voxels borde, y a continuación refinar la localización para obtener una resolución a nivel subvoxel.

En cuanto a la detección a nivel voxel, la primera extensión significativa desde 2D a 3D fue realizada por Zhang [ZHA93]. Como método más simple, el autor sugería aplicar una máscara 2D a los voxels en las tres direcciones para determinar las tres derivadas parciales, aunque también introdujo la aplicación de máscaras 3D completas. Bhattacharya y Wild [BHA96] dedujeron un detector de borde 3D al estilo del detector de Sobel. Mehrotra y Zhan [MEH94] derivaron un detector basado en los cruces por cero en 3D (*zero-crossing*), usando el mismo criterio de optimalidad del detector de Canny. Posteriormente, Deriche [DER90] propuso un nuevo método para implementar una variación del filtro de Canny en 2D de una forma recursiva mucho más rápida, y Monga [MON91] realizó la extensión para imágenes 3D.

En cuanto a la detección a nivel subvoxel, de nuevo nos encontramos con la extrapolación de las técnicas anteriores para 2D, aunque hay que tener en cuenta que el coste computacional de todos estos métodos al pasar a tres dimensiones es bastante alto. Las más usadas siguen siendo las técnicas de interpolación, que en el caso de 3D, consisten básicamente en interpolar la función gradiente en el interior de una vecindad del máximo local

usando un modelo conveniente, y finalmente asumir que la superficie del borde viene dada por una superficie de nivel (*isosuperficie*) de dicha función.

Existen numerosos métodos en la literatura para extraer isosuperficies. La solución más clásica es el algoritmo Marching Cubes [LOR87], optimizado posteriormente por Nielson y Hamann [NIE91] para eliminar las ambigüedades producidas por el método original en cierto tipo de superficies. Por otro lado, Monga y Benayount [MON95] proponen extraer la superficie a partir de una estimación de las derivadas parciales de la imagen. Otras técnicas más elaboradas son las siguientes: Itoh y Koyamada [ITO95] usan estructuras de datos complejas como grafos extremos y listas ordenadas de pixels frontera para propagar una isosuperficie de forma eficiente. Livnat y otros [LIV96] proponen un espacio de segmentos (*span space*) y el uso de *árboles-kd* para reducir el tiempo de búsqueda, proyectando los datos sobre un espacio de este tipo. Thirion y Gourdon [THI96] proponen el algoritmo de Marching Lines para extraer líneas 3D correspondientes a la intersección de dos isosuperficies.

En cuanto a las técnicas basadas en el cálculo de momentos, Luo y otros [LUO93] proponen un conjunto de máscaras 3D para estimar los momentos de nivel de gris mediante la correlación con la imagen. Cada máscara se calcula en el interior de una esfera con un cierto radio. Hamitouche y otros [HAM95] proponen una variación para tomar en cuenta la anisotropía de los valores debido a las diferentes tamaños de muestreo en cada dirección del espacio 3D. Ibáñez y otros [IBA96] proponen un operador basado en el cálculo de momentos usando tres perfiles diferentes de borde (constante, lineal y cuadrático) para detectar directamente la superficie con precisión sub-voxel.

Otras alternativas diferentes son: Tang y Medioni [TAN98] proponen una metodología para procesar la imagen mediante una técnica llamada *TensorVoting*, la cual produce campos vectoriales densos, que a su vez sirven de entrada para una segunda técnica llamada *Algoritmo de Superficies Extremas*, el cual extrae con precisión sub-voxel los parámetros característicos que representan los extremos locales de dicho campo vectorial. Por otro lado, Hwang y Wehrli [HWA02] proponen una etapa de preproceso llamada *Subvoxel Processing* para reducir el difuminado de los bordes generado en la adquisición de la imagen, incrementando la resolución de la misma, y así detectar el borde en la nueva imagen.

0.3.3 Detección de bordes en angiografías

La mayoría de las técnicas de detección usadas en angiografías se basan en métodos generalistas como los comentados anteriormente, principalmente las basadas en el cálculo de las derivadas. Así, muchos autores (ver [REI84, SPE83a]) aplican métodos de cálculo de la primera y segunda derivada a lo largo de la línea perperdicular a lo que generalmente se conoce como *centerline* (línea continua que indica los puntos interiores centrales del vaso). Otros autores como Koojiman [KOO82] intentan ser más precisos, y para ello calculan además la recta de regresión lineal a ambos lados del borde con el fin de eliminar la contribución de la intensidad de fondo en los pixels borde.

Existen varios problemas en este tipo de imágenes: el desenfoque y el ruido pueden producir discontinuidad en los bordes detectados, por lo que se suele usarse alguna etapa de filtrado previo. Por otro lado, los vasos demasiado estrechos producen valores erróneos en la posición de los bordes, por lo que suele suavizar su perfil para ajustarlo a una curva antes de aplicar el detector. Otro factor importante es el tamaño de la ventana donde se realiza el cálculo de las derivadas: si es muy pequeño aparecerán bordes no esperados, y si es muy grande se pierde precisión en la localización. Por ello algunos autores proponen un tamaño de ventana ajustable empíricamente [KIR82, SPE83b].

Como ya hemos comentado, el objetivo de la presente tesis se centra en localizar con precisión los parámetros asociados al borde en el interior de cada pixel. Sin embargo, existen otros trabajos que van más alla, y tratan de detectar el perfil continuo de todo el vaso. A continuación se destacan algunos de ellos, tanto para imágenes 2D como 3D.

Detección avanzada en angiografías 2D

Muchos autores han propuesto técnicas automáticas para la delineación de los vasos. Fukui [FUK80] desarrolló un algoritmo automático basado en las características del árbol de vasos (*arterial tree*), el cual primero detecta segmentos candidatos para ser parte de los bordes de los vasos, y luego los usa para ir caminando por ellos mediante un algoritmo de tracking. Shumeli [SHU83] presentó una técnica para calcular las fronteras y el diámetro de una arteria, basándonse en un modelo estocástico y asumiendo secciónes circulares para los vasos. Nguyen y Sklansky [NGU86] propusieron un algoritmo de detección, asumiendo un modelo elíptico para la sección del vaso, el cual busca primero la línea de máximos en la primera derivada a lo largo del vaso usando ventanas pequeñas, y luego estima las fronteras aplicando el método de máxima pendiente en la línea perpendicular a la línea de máximos. Eichel y otros [EIC88] presentaron un algoritmo de detección y concatenación de bordes basado en un modelo de borde Markoviano, el cual asume intensidad uniforme en el exterior del vaso, lo que lo hace bastante sensible según el tipo de imagen. Sonka y otros [SON95] desarrollaron un método de detección de bordes simultáneos basándose en una función de coste, con excelentes resultados en imágenes complejas pero sin demasiado ruido.

Otra clase de técnicas usan una aproximación paramétrica, basada en un conocimiento a priori del perfil de la arteria. Estos métodos suelen dar mejores resultados que los basados en el cálculo de derivadas, puesto que tienen en cuenta la textura del fondo y la degradación introducida por el sistema de adquisición. Por ejemplo, para modelar la textura de huesos y tejido orgánico, Kitamura [KIT88] utiliza polinomios de primer orden, mientras que Pappas [PAP88] los utiliza de orden más alto. Otra gran ventaja del modelado paramétrico es que las medidas significativas, como pueden ser el diámetro del vaso o el área de la sección, pueden calcularse fácilmente a partir de los parámetros estimados para el vaso. Para estimar los bordes, se considera una malla de posibles valores para los parámetros, y se selecciona la que dé mínimo error entre los datos estimados y observados en la imagen (ver [KAY93, CHA00]).

Una clase diferente de aproximación es la que usa filtros. Sun [SUN89] propuso un algoritmo de tracking adaptativo empleando la continuidad espacial del *centerline* del vaso, así como de su orientación, diámetro y densidad, aunque no funciona muy bien para fondos con intensidad no uniforme. Klein y otros [KLE97] introdujeron un método que utiliza modelos de spline deformables y filtros de Gabor para detectar los bordes de forma eficiente cuando la intensidad del fondo no es uniforme. Van der Zwet y otros [VAN98] aplican la teoría de Canny añadiendo restricciones adicionales a las fronteras. Poli y Valli [POL97] diseñaron un algoritmo para detectar y realzar los vasos en tiempo real, basándose en un conjunto de filtros lineales sensibles a las diferentes orientaciones y grosores de los vasos.

Otras técnicas diferentes son la propuesta por Weber [WEB89], que detecta los bordes usando los cruces por cero de la transformada de Fourier de la imagen, o la introducida por Kayikcioglu y otros [KAY02], donde se propone un método que, a partir de la modelización de los vasos como cilindros generalizados de base elíptica, desarrolla una superficie 2D para la distribución de la intensidad en una zona del vaso, detectando simultáneamente los bordes izquierdo y derecho de éste.

Detección avanzada en angiografías 3D

El principal objetivo en las angiografías 3D es poder detectar vasos de distintos tamaños, y una manera de lograrlo es a través del análisis multiescala. Así, Koller y otros [KOL95] proponen una respuesta multiescala con el fin de detectar estructuras lineales, la cual consigue discriminar entre contornos y *centerlines*. Para ello toman como información los autovectores de la matriz Hessiana, y logran definir en cada punto la dirección ortogonal al eje central del posible vaso que pasa por dicho punto. Continuando con este trabajo, Lorenz y otros [LOR97] decidieron usar además la información de los autovalores de dicha matriz, y en función de la magnitud de éstos poder diferenciar entre estructuras tubulares como los vasos u otras estructuras diferentes. En esta misma línea, Sato y otros [SAT97, SAT98] usan la heurística para elegir la función de respuesta que combina los tres autovalores, basándose en un estudio experimental de varios casos, mientras que Frangi y otros [FRA98]

utilizan una interpretación geométrica de los autovalores para elegir la función de respuesta.

Otro trabajo importante en esta misma línea es el de Krissian y otros [KRI00c], los cuales proponen una nueva aproximación para detectar la línea central (*centerline*) de los vasos y reconstruir su geometría tridimensional. Para ello introducen varios tipos de modelos de vaso, con el fin de estimar las derivadas de segundo orden de la imagen en función de la sección elíptica de los vasos, la curvatura del eje, o los efectos de difuminado de la intensidad en los voxels. De esta forma, para cada modelo de vaso se deriva una expresión analítica de la relación entre el radio de la estructura y la escala a la que fue detectada, lo que permite estimar con precisión la localización del vaso y su grosor.

Existen también otros trabajos que no usan este tipo de análisis, como son el de Verdonck [VER96], que ajusta un modelo de contorno de los vasos a la imagen, el de Szekely y otros [SZE94], que proponen una descripción simbólica de la red de vasos y muestran los resultados en angiografías de resonancia magnética (MRA), o el de Summers y otros [SUM97] que usan una estructura de datos multiresolución para extraer la morfología vascular y los parámetros locales a partir de imágenes contrastadas.

0.4 Aportaciones originales

Centrándonos primeramente en la detección en imágenes bidimensionales, las aportaciones principales de este trabajo son las siguientes:

- **Localización de contornos de grado 1 a nivel sub-pixel.** A partir de un modelo de borde y otro de adquisición que plantearemos al principio, deduciremos un primer esquema de detección de contornos a nivel sub-pixel, asumiendo que localmente en el interior de cada pixel, el contorno puede ser aproximado por un segmento recto. Dicho detector, partiendo únicamente de los valores de intensidad de una vecindad alrededor de cada pixel, logrará estimar con gran exactitud, al menos para imágenes ideales, los parámetros característicos del contorno en el interior de dicho pixel, a saber, localización, orientación, y diferencia de intensidad a ambos lados del contorno.

- **Localización de contornos de grado 2 a nivel sub-pixel.** Modificando la aproximación lineal del contorno por una aproximación cuadrática, obtendremos un detector de grado 2 que buscará para cada pixel el arco de parábola que mejor se ajuste al contorno. De esta forma, se aumenta la precisión del método, y se obtiene un valor adicional: la curvatura del contorno en dicho pixel.

- **Localización de contornos de grado 1 a nivel sub-pixel en imágenes previamente suavizadas.** Todo proceso de adquisición introduce ruido en la imagen, lo cual distorsiona el valor de los pixels. Esta distorsión provocará que el proceso de detección obtenga resultados erróneos. Para tratar de disminuir el ruido se aplica una etapa de suavizado. Por lo tanto, deduciremos un nuevo esquema de detección de contornos que, partiendo únicamente de los valores de intensidad de la imagen suavizada, logre estimar con gran exactitud, al menos para imágenes ideales, los parámetros del contorno presente en la imagen original antes de ser suavizada.

- **Localización de contornos de grado 2 a nivel sub-pixel en imágenes previamente suavizadas.** Se procede igual pero modificando la aproximación inicial por una cuadrática.

- **Localización de contornos de grado 1 a nivel sub-pixel en imágenes con alto nivel de ruido.** Cuando el ruido es elevado, un simple suavizado no basta para eliminarlo del todo. Para resolver el problema de la detección en este tipo de imágenes propondremos un esquema de restauración que vaya mejorando la calidad de la imagen en cada iteración, con el fin de aplicar nuestro detector anterior sobre la imagen restaurada final y obtener con precisión los contornos presentes en la imagen original. Dicho esquema consistirá de tres etapas en cada iteración: un suavizado inicial, un proceso de detección de

contornos, y la generación de una nueva imagen sintética a partir de los valores obtenidos por el método detector.

- **Localización de contornos de grado 2 a nivel sub-pixel en imágenes con alto nivel de ruido.** Para poder obtener un esquema de mayor precisión y que sea lo más estable posible a lo largo de las iteraciones, no es suficiente sustituir la aproximación lineal por una que utilice arcos de parábolas. Para lograrlo se propondrá un primer esquema que aproxime localmente el contorno por un arco de circunferencia. Posteriormente, desarrollaremos un segundo esquema de restauración y detección mejorado para poder resolver dos casos diferentes que causan problemas en el esquema inicial. Estos casos son contornos que posean un valor de curvatura muy alto, y contornos que se encuentren muy cercanos entre sí, situados a pocos pixels de distancia.

En cuanto a la detección en las imágenes tridimensionales, las aportaciones principales del trabajo vuelven a ser las mismas, pero adaptadas al nuevo tipo de imagen:

- **Localización de contornos de grado 1 a nivel sub-voxel.** Extrapolando el modelo de borde y adquisición a tres dimensiones, deduciremos un esquema de detección de contornos a nivel sub-voxel, asumiendo que localmente en el interior de cada voxel, el contorno puede ser aproximado por un plano. Dicho detector, partiendo también únicamente de los valores de intensidad alrededor de cada voxel, logrará estimar con gran exactitud, al menos para imágenes ideales, los parámetros característicos del contorno en el interior de dicho voxel, como son su localización, el salto de intensidad a ambos lados del contorno, y la orientación, la cual vendrá dada por un vector normal a la superficie.

- **Localización de contornos de grado 2 a nivel sub-voxel.** Modificando la aproximación lineal del contorno por una aproximación cuadrática, obtendremos un detector de grado 2 que buscará para cada voxel el paraboloide que mejor se ajuste al contorno. De esta forma, se aumenta la precisión del método, y se obtienen nuevos valores adicionales para cada voxel: por un lado, los valores de curvatura mínima y máxima de la superficie en dicho voxel, y por otro lado las direcciones en las que ambas se producen. Estos dos vectores junto con el vector normal formaran un sistema ortonormal que definirá de manera precisa la geometría de la superficie en el interior del voxel.

- **Localización de contornos de grado 1 a nivel sub-voxel en imágenes previamente suavizadas.** El proceso de adquisición de las imágenes 3D también introduce ruido en los valores de intensidad, por lo que de nuevo es necesaria una etapa previa de suavizado. Por lo tanto, deduciremos un nuevo esquema de detección de contornos que, partiendo únicamente de los valores de intensidad de la imagen suavizada, logre estimar con gran exactitud, al menos para imágenes ideales, los parámetros del contorno presente en la imagen original antes de ser suavizada.

- **Localización de contornos de grado 2 a nivel sub-voxel en imágenes previamente suavizadas.** Se procede igual pero modificando la aproximación inicial por una cuadrática.

- **Localización de contornos de grado 1 a nivel sub-voxel en imágenes con alto nivel de ruido.** Para realizar la detección en imágenes con alto nivel de ruido propondremos, al igual que en caso 2D, un esquema inicial de restauración, con el fin de aplicar nuestro detector anterior sobre la imagen restaurada final y obtener con precisión los contornos presentes en la imagen original.

- **Localización de contornos de grado 2 a nivel sub-voxel en imágenes con alto nivel de ruido.** Al igual que ocurría en el caso 2D, la aproximación por paraboloides simples no es suficiente para lograr un esquema de restauración estable, por lo que se utilizará una aproximación algo más compleja utilizando paraboloides rotados. Posteriormente, mejoraremos el esquema de restauración y detección para poder resolver los casos conflictivos de altas curvaturas y superficies muy cercanas entre sí.

0.5 Software desarrollado

Simultáneamente al estudio de los métodos expuestos en esta tesis, se ha tenido que desarrollar software que permita la implementación de los esquemas que se plantean, así como la visualización de los resultados. Aunque la tarea de programación suele obviarse en muchos artículos y textos, quedando relegada casi siempre a un segundo plano frente a los conceptos y técnicas matemáticas expuestas, consideramos necesario al menos hacer mención de las aplicaciones desarrolladas durante el tiempo que ha durado esta investigación.

¿Por qué construir nuestro propio software, en lugar de utilizar alguno de los ya existentes? En [CHR94] puede verse una descripción de los más importantes, aunque una lista más actualizada puede verse en [1]. Es cierto que existen bastantes hoy en día, pero al comienzo de nuestra investigación[2] no eran tantos, y muchos de ellos eran o siguen siendo de pago, o en la modalidad shareware, y había que pagar para tener toda su funcionalidad. Además, no todos estaban orientados a la programación de nuevos algoritmos, sino a ejecutar los ya existentes.

Quizás el más extendido por ese entonces fuera el sistema llamado **Khoros** [KON94, KHO97], desarrollado en la Universidad de Nuevo México[3]. Incluía un enorme conjunto de técnicas clásicas de tratamiento de imágenes, y poseía una interfaz visual que permitía la interconexión de módulos para formar sistemas completos. Permitía además definir estructuras de control simples para realizar experimentos y procesos repetitivos, y disponía de una serie de herramientas para el diseño de nuevas funciones e interfaces de usuario. Aun así, Khoros no era muy sencillo de usar, por lo que aquellos investigadores sin mucho conocimiento de informática necesitaban dedicar mucho tiempo para hacerse con el sistema antes de empezar a trabajar con él.

Hay que tener en cuenta que en el área del tratamiento de imágenes no sólo trabajan informáticos, sino también investigadores pertenecientes a otras áreas, como física, matemáticas, biología, medicina, y cualquier otra materia en donde se necesite procesar imágenes para obtener algunos resultados a partir de ellas. Para estos investigadores, la mayor parte de los entornos de tratamiento de imágenes en ese entonces eran complejos de entender y usar por completo. Además, el investigador que deseara desarrollar su propio algoritmo de tratamiento de imágenes, tenía normalmente que escribir código también para leer y guardar imágenes en disco, mostrarlas en pantalla, calcular sus histogramas, convolucionar con máscaras, manejar los eventos del teclado y del ratón, etc. Normalmente, todas estas rutinas solían ocupar un porcentaje muy elevado de la cantidad de código total que debía escribirse.

El objetivo era por tanto doble: por un lado desarrollar un software de libre distribución que automatizara toda esta parte, permitiendo al investigador concentrarse en el código de su algoritmo, y por otro lado conseguir que todo el trabajo de programación desarrollado en nuestra investigación fuese aprovechable para una comunidad científica más amplia. De esta forma se lograría minimizar el tiempo necesario para que cualquier tipo de investigador pueda implementar y depurar su propio algoritmo, y poder entonces obtener resultados de forma más rápida. Así nació XMegaWave.

0.5.1 XMegaWave

XMegaWave (XMW) es un entorno gráfico de ventanas, de libre distribución[4], orientado al tratamiento de imágenes. El desarrollo del XMW ha sido posible gracias a la colaboración entre investigadores de la Universidad de las Islas Baleares, la Universidad de Las Palmas de Gran Canaria y la Universidad de Paris IX Dauphine, que empezaron su desarrollo en el año 1992. Desde entonces ha sido expuesto en varios congresos nacionales e internacionales, así como en varias universidades en donde este investigador ha realizado estancias [TRU94, TRU98a, TRU98b]. En la figura 8 puede verse su apariencia.

[1] http://www-2.cs.cmu.edu/afs/cs/project/cil/ftp/html/v-source.html

[2] Hablamos del año 1992, fecha en la que nuestro grupo de investigación comenzó su andadura.

[3] Aunque comenzó siendo de libre distribución, hoy en día ha derivado en el software comercial VisiQuest, propiedad de la compañía AccuSoft.

[4] http://amiserver.dis.ulpgc.es/xmwgus

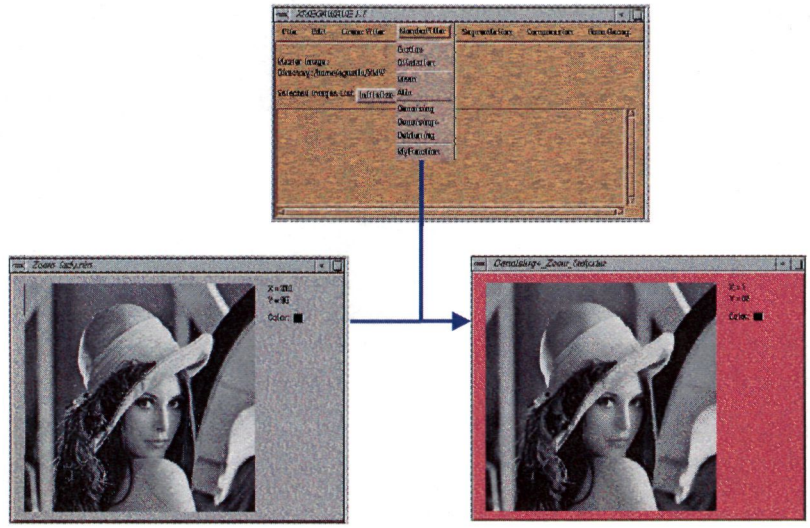


Figura 8: Ejemplo de ejecución de un algoritmo de tratamiento de imagen utilizando XMW

Sus características principales son:

- XMW es un **entorno de tratamiento de imágenes**, con una interface de ventanas, que funciona en estaciones Unix o Linux. Por un lado es similar a las aplicaciones típicas de dibujo, con opciones diferentes para abrir y guardar imágenes en varios formatos, y dibujar primitivas geométricas. Además puede trabajar con secuencias de imágenes (vídeo). Por otro lado, es un programa de retoque fotográfico, donde pueden utilizarse las típicas técnicas de tratamiento de imagen (llamadas comúnmente *filtros*).

- XMW es una **librería de tratamiento de imágenes**, que incluye técnicas clásicas como balance del contraste, adición de ruido blanco, promediado, suavizado gaussiano, detector de bordes, etc. Además, incluye varias técnicas de análisis multiescala, como los filtros de erosión y dilatación, deblurring, denoising, curvatura media, etc.

- XMW es una **librería de programación**, donde el usuario puede implementar su propio algoritmo de tratamiento de imágenes en lenguaje C usando la librería de forma muy fácil. Por un lado, la librería incluye funciones para abrir y guardar imágenes, mostrarlas en la pantalla, dibujar sus histogramas, etc. Por otro lado, todos los filtros mencionados anteriormente son accesibles desde código, por lo que nuestro algoritmo puede usarlos como una fase previa o posterior. Estas funciones desarrolladas por los usuarios pasan a forma parte del núcleo de XMW, por lo que el entorno puede crecer a medida que los usuarios desarrollan nuevos filtros.

- XMW es una **herramienta pedagógica** destinada a usarse en clases de tratamiento de imágenes. Su funcionamiento puede explicarse en una sola clase, y los estudiantes pueden rápidamente implementar cualquier tipo de filtro y ver inmediatamente sus resultados. La cantidad necesaria de código que el estudiante debe escribir es realmente pequeña, y el tiempo necesario para depurar es muy corto, ya que no hay que preocuparse por cómo mostrar las imágenes en la pantalla, guardarlas en el disco, calcular histogramas, etc. Además tampoco se necesitan conocimientos sobre librerías complejas de programación de ventanas típicas de sistemas Unix (X11, Motif, OpenLook, etc.).

Aunque ya han transcurrido muchos años desde la primera versión, y sus características ya no resulten tan novedosas como entonces, las nuevas versiones han ido incorporando todas aquellas utilidades, tanto de visualización como de ayuda al desarrollo de filtros y a su depuración, que se han ido necesitando, a medida que la investigación ha ido avanzando.

0.5.2 AMILab

Originalmente el XMW estaba orientado sólamente a imágenes 2D. Aprovechando la estancia del investigador Karl Krissian en nuestro laboratorio de investigación durante los años 2000 y 2001, el cual había comenzado a desarrollar varias aplicaciones para tratamiento de imagen en 3D, se generó un nuevo software, uniendo lo mejor de ambos códigos, para generar un nuevo entorno de desarrollo orientado al tratamiento de imágenes 2D y 3D llamado AMILab [TRU01]. En la figura 9 puede verse un ejemplo de utilización.

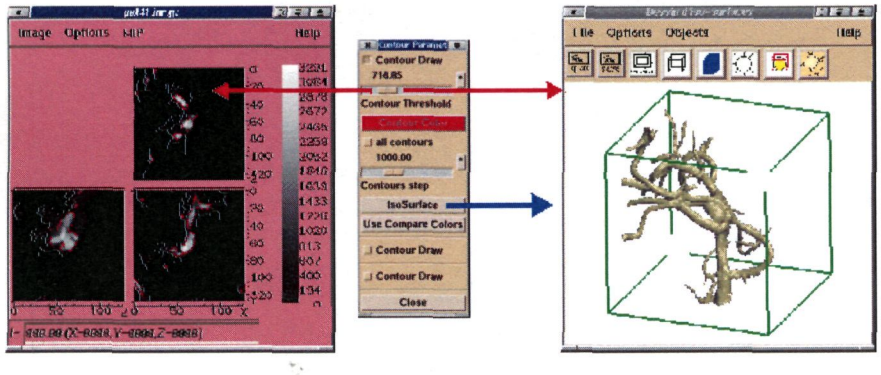


Figura 9: Ejemplo de visualización de una isosuperficie utilizando AMILab

Este nuevo entorno, también de libre distribución[5], añadía nuevas características a las anteriores que poseía XMW:

- Todos los filtros incluidos en el núcleo de XMW fueron extrapolados a tres dimensiones.
- Se añadieron nuevos formatos gráficos para la lectura y escritura de las imágenes al disco.
- Se permitió el uso del lenguaje C++ para el desarrollo de los filtros, ya que XMW sólo permitia el uso de lenguaje C.
- Se creó una **herramienta de visualización 3D** para poder mostrar las imágenes tridimensionales de varias maneras: superficies de nivel, voxels, secciones, proyección de la intensidad máxima (MIP), etc. Los parámetros de dichos algoritmos están disponibles desde los menús y las barras de botones de la aplicación. Además estos algoritmos pueden utilizarse como ficheros ejecutables independientes, sin tener que cargar la aplicación completa en memoria
- Se integró un **intérprete** dentro de la aplicación, de forma que puedan aplicarse todos los filtros y demás herramientas a la imagen desde una ventana de línea de comandos, lo cual agiliza enormemente el tiempo de depuración, así como facilita la creación de demos.

[5] http://amiserver.dis.ulpgc.es/amilab

0.6 Estructura del documento

La parte principal del trabajo se encuentra dividida en diez capítulos, los cuales están agrupados en 2 bloques claramente diferenciados, uno para imágenes bidimensionales y otro para imágenes tridimensionales. Cada capítulo trata de resolver un problema diferente. En cada uno se comienza exponiendo una breve discusión sobre las técnicas tradicionales para resolver cada problema, a continuación se desarrolla y se presenta la solución propuesta, y finalmente se aplica a varias imágenes ejemplo, realizando en algunos casos comparativas con otros métodos. Además, los capítulos están escritos siguiendo un orden cronológico, de tal forma que su lectura secuencial representa una memoria del transcurso de la investigación durante los últimos años.

El primer bloque está dedicado al estudio de la detección en imágenes bidimensionales. En el capítulo 1 se comentan las técnicas tradicionales para la detección de bordes, y analizamos en profundidad el error que cometen incluso en imágenes ideales, debido a la naturaleza discreta de los pixels de la imagen. En los capítulos 2 y 3 planteamos los primeros esquemas de detección de contornos a nivel sub-pixel de grado 1 y 2 respectivamente, aproximando el contorno localmente como un segmento recto o parábolico según cada caso. En el capítulo 4 desarrollamos un nuevo método que, partiendo de una imagen suavizada, permita detectar los contornos presentes en la imagen original. Finalmente, en los capítulos 5 y 6 se plantean dos esquemas de restauración: uno inicial para casos ideales, donde los contornos se encuentren aislados, y otro mejorado que resuelva la detección también en casos donde existan contornos con alta curvatura, o contornos muy cercanos entre sí.

El segundo bloque está dedicado a las imágenes tridimensionales. En los capítulos 7 y 8 se extrapolan las ideas y técnicas del bloque anterior para deducir un primer esquema de detección de contornos a nivel sub-voxel de grado 1 y 2 respectivamente. En el capítulo 9 se desarrolla un método de detección para su aplicación en imágenes previamente suavizadas. Finalmente en el capítulo 10 se proponen los esquemas de restauración.

Por último existe un tercer bloque, en donde se incluyen las conclusiones obtenidas en la investigación y los trabajos futuros que quedarían abiertos, así como varios anexos en donde se muestran algunos desarrollos matemáticos y algorítmicos que han sido utilizados y que se ha creido conveniente no incluir dentro de los dos bloques principales.

0.7 Otras consideraciones

- Todo el trabajo de investigación y desarrollo se ha realizado en el laboratorio del grupo de investigación AMI (Análisis Matemático de Imágenes) del Departamento de Informática y Sistemas de la Universidad de Las Palmas de Gran Canaria.

- La mayoría de las imágenes médicas han sido cedidas por cortesía del investigador Karl Krissian, tras su paso por el Instituto Francés de Investigación en Informática y Automática (INRIA). Las imágenes finales del capítulo 10 han sido cedidas por el profesor Vaclav Skala, del Centro de Gráficos por Computador y Visualización de Datos de la Universidad de West Bohemia en Pilsen (República Checa).

- El último año de esta investigación ha sido financiado por la Fundación Canaria Universitaria de Las Palmas, a través de una ayuda Innova de 3.000 euros aportados por la empresa Beleyma.

Parte I

Localización de contornos en imágenes 2D

Capítulo 1

Técnicas convencionales para el cálculo del gradiente

- Luis: ¿te has dado cuenta que el cálculo de la orientación de los bordes, usando las técnicas clásicas, no es exacto?
- Es lógico, teniendo en cuenta que sólo disponemos de valores discretos en la imagen.
- Aún así, ni siquiera en imágenes ideales....

SEPTIEMBRE 2000

Tradicionalmente se ha venido interpretando un borde en una imagen como un salto brusco de la intensidad. Si representamos la imagen como una función bidimensional que mide la intensidad en cada punto de la imagen, los bordes corresponderán a zonas donde la intensidad varíe rápidamente. La técnica más básica que suele usarse para detectar bordes es calcular la derivada de la función y buscar máximos locales en la dirección del gradiente. Como se sabe, la derivada de una señal continua proporciona las variaciones locales con respecto a la variable, de tal forma que el valor de la derivada será mayor cuanto más rápidas son estas variaciones. Esta idea fue sugerida inicialmente, tanto desde un punto de vista biológico como computacional, por Marr [MAR82], y ha sido la referencia para la mayoría de los trabajos posteriores.

Realmente las técnicas de detección de bordes son esquemas más sofisticados que un simple cálculo de derivadas, pero este cálculo es parte sustancial de casi todas estas técnicas. La estimación del vector gradiente en un pixel de la imagen es un cálculo que prácticamente todas las técnicas precisan. En este capítulo analizaremos en profundidad el funcionamiento de este esquema de cálculo del gradiente, y mostraremos el error cometido en algunos casos.

1.1 Detección de bordes a partir del cálculo del vector gradiente

Para poder calcular el vector gradiente en un pixel, es preciso primero poder estimar el valor de las dos derivadas, horizontal y vertical, en dicho pixel. Además, este cálculo debe hacerse de forma discreta, ya que la imagen está discretizada. A continuación analizaremos primero el comportamiento de la derivada sobre un borde, y luego deduciremos el esquema tradicional que suele usarse para poder discretizar su expresión.

1.1.1 La derivada como medida de los cambios de intensidad

Tomemos primero el caso unidimensional. Sea $f(x)$ una señal unidimensional que represente la intensidad que hay en el punto x. La imagen de un borde será algo similar a la figura 1.1, donde alrededor del punto x_0 se ha producido un salto brusco de la intensidad. Este hecho queda reflejado en la derivada de la señal, que alcanza un máximo local en ese punto.

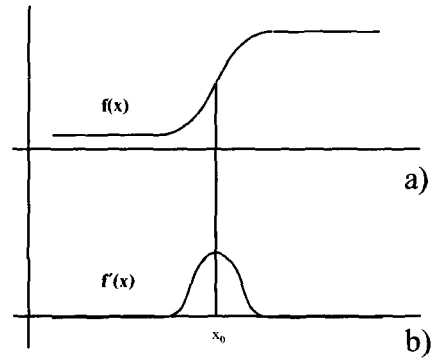


Figura 1.1: Representación unidimensional de un borde: a) Señal original. b) Derivada.

En el caso bidimensional, el fenómeno es muy parecido: un borde también viene dado por un salto brusco de la intensidad. Por ejemplo, si tuviésemos la imagen de un círculo claro sobre fondo oscuro como el de la figura 1.2a, esto podría interpretarse como que tenemos una función de dos variables, $f(x, y)$, cuyo valor representa la intensidad del punto (x, y), tal y como se ve en la figura 1.2b.

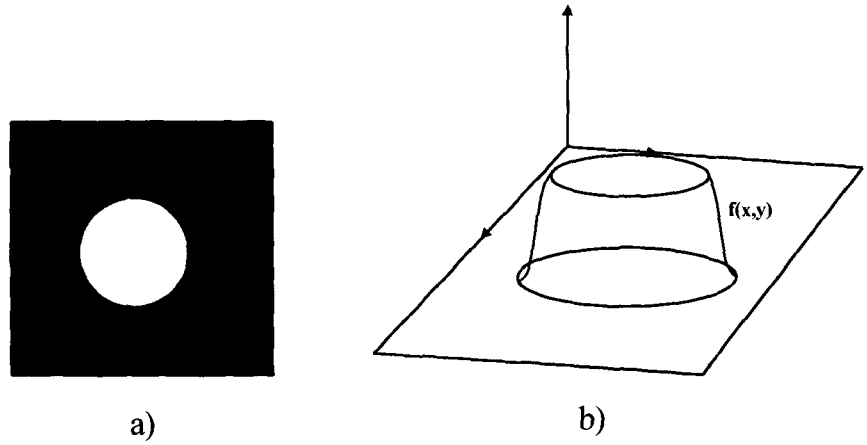


Figura 1.2: Representacion bidimensional de un borde: a) Valores de intensidad. b) La imagen vista como una superficie.

Para una función de dos variables, el vector gradiente viene dado por las derivadas parciales de la función:

$$\nabla f(x,y) = \left[\frac{\partial f(x,y)}{\partial x}, \frac{\partial f(x,y)}{\partial y}\right]$$

Éste es un vector bidimensional cuya orientación nos indica la dirección de máxima variación en cada punto, y cuyo módulo representa la magnitud de dicha variación. Esto significa que el vector gradiente tendrá magnitud pequeña sobre los puntos de las zonas homogéneas clara y oscura, y tendrá un valor muy alto justo en la zona donde se produce la transición de una intensidad a otra. Además, la dirección de dicho vector nos indicará en cada punto en qué dirección se produce ese cambio. En el ejemplo que nos ocupa, el vector apuntaría siempre hacia el centro del círculo.

Por lo tanto, un proceso básico en el tratamiento de imágenes consiste en evaluar el vector gradiente en cada pixel, y quedarnos con aquéllos cuyas magnitudes sean máximos locales en la dirección de dicho vector, tal y como proponen Marr y Hildreth [MAR80], Canny [CAN83, CAN86], y muchos otros [DER87, FLE92]. Pero para poder evaluar el gradiente, necesitamos conocer cómo discretizar las derivadas, ya que nuestra imagen de entrada no será una señal continua sino discreta, debido al proceso de adquisición mediante un dispositivo fotográfico.

1.1.2 Discretización de la derivada de una función $f(x)$

Sea $f(x)$ una función continua de $\mathbb{R}$ en $\mathbb{R}$. Para estimar el valor de la primera derivada en un punto x_0 a partir del valor de la función en puntos cercanos, podemos usar el desarrollo de Taylor de la función en los puntos $(x_0 + h)$ y $(x_0 - h)$, de la siguiente manera:

$$\begin{aligned} f(x_0 + h) &= f(x_0) + hf'(x_0) + O(h^2) \\ f(x_0 - h) &= f(x_0) - hf'(x_0) + O(h^2) \end{aligned}$$

donde h es un valor relativamente pequeño (cuanto menor sea, mejor será la aproximación). Restando ambas expresiones obtenemos que

$$f(x_0 + h) - f(x_0 - h) = 2hf'(x_0) + O(h^2)$$

de donde se deduce que, para valores pequeños de h, la primera derivada en el punto x_0 puede aproximarse por

$$f'(x_0) \simeq \frac{f(x_0 + h) - f(x_0 - h)}{2h}$$

Esto significa que, si la función $f(x)$ ha sido discretizada en una muestra $\{..., f(x_{i-1}), f(x_i), f(x_{i+1}), ...\}$, $i \in Z$, donde la distancia horizontal h entre puntos vecinos puede considerarse pequeña (fig.1.3), y de tal forma que desconocemos los valores de la función en ningún otro lugar, podemos estimar la derivada en el punto x_i a partir de los valores de la función en sus dos puntos vecinos, x_{i-1} y x_{i+1}, mediante la siguiente expresión:

$$f'(x_i) \simeq \frac{f(x_{i+1}) - f(x_{i-1})}{2h}$$

1.1.3 Discretización de la derivada parcial de una función $f(x,y)$.

Sea ahora $f(x,y)$ una función de $\mathbb{R}^2$ en $\mathbb{R}$. Para estimar el valor de la derivada parcial en la dirección de la variable x en un punto (x_0, y_0) a partir del valor de la función en puntos cercanos, podemos usar el desarrollo de Taylor de la función en los puntos $(x_0 + h, y_0)$ y $(x_0 - h, y_0)$, de la siguiente manera:

$$\begin{aligned} f(x_0 + h, y_0) &= f(x_0, y_0) + hf_x(x_0, y_0) + O(h^2) \\ f(x_0 - h, y_0) &= f(x_0, y_0) - hf_x(x_0, y_0) + O(h^2) \end{aligned}$$

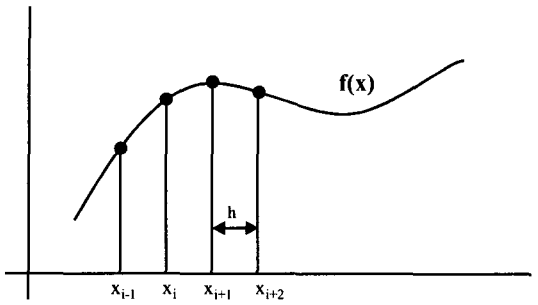


Figura 1.3: Función unidimensional discretizada

donde h es un valor relativamente pequeño. Restando ambas expresiones obtenemos que

$$f(x_0 + h, y_0) - f(x_0 - h, y_0) = 2hf_x(x_0, y_0) + O(h^2)$$

de donde se deduce que, para valores pequeños de h, la derivada parcial en x en el punto (x_0, y_0) puede aproximarse por

$$f_x(x_0, y_0) \simeq \frac{f(x_0 + h, y_0) - f(x_0 - h, y_0)}{2h}$$

Al igual que ocurría con las funciones de una variable, esto significa que, si la función $f(x, y)$ ha sido discretizada en una muestra bidimensional $\{..., f(x_{i-1}, y_j), f(x_i, y_j), f(x_{i+1}, y_j), ..., f(x_{i-1}, y_{j+1}), f(x_i, y_{j+1}), f(x_{i+1}, y_j), ...\}$, $(i, j \in Z)$, donde la distancia horizontal h entre puntos vecinos puede considerarse pequeña (fig.1.4), y de tal forma que desconocemos los valores de la función en ningún otro lugar, podemos estimar la derivada parcial en la dirección x en el punto (x_i, y_j) a partir de los valores de la función en sus dos puntos vecinos, (x_{i-1}, y_j) y (x_{i+1}, y_j), mediante la siguiente expresión:

$$f_x(x_i, y_j) \simeq \frac{f(x_{i+1}, y_j) - f(x_{i-1}, y_j)}{2h}$$

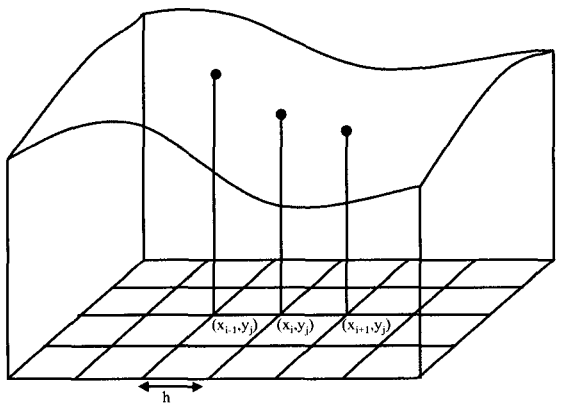


Figura 1.4: Función bidimensional discretizada

Lógicamente, al haber discretizado la función, esto no es más que una estimación del verdadero valor de la derivada. Otra estimación alternativa que puede dar mejores resultados es usar también la anterior expresión pero aplicada a los puntos vecinos (x_i, y_{j-1}) y (x_i, y_{j+1}), es decir, en las filas anterior y posterior al punto que estamos tratando. De esta manera, el valor final que daremos para la derivada parcial en la dirección x en el punto (x_i, y_j) será un promedio entre las expresiones individuales de las tres filas:

$$\begin{aligned} f_x(x_i, y_j) &= (1-\alpha)\left(\frac{f(x_{i+1}, y_j) - f(x_{i-1}, y_j)}{2h}\right) + \\ &+\frac{\alpha}{2}\left(\frac{f(x_{i+1}, y_{j-1}) - f(x_{i-1}, y_{j-1}) + f(x_{i+1}, y_{j+1}) - f(x_{i-1}, y_{j+1})}{2h}\right) \end{aligned} \tag{1.1}$$

donde $0 \leqslant \alpha \leqslant 1$. Con el valor α controlamos el peso que queremos darle a las filas adyacentes. Si $\alpha = 0$ sólo consideramos la fila actual, y estaríamos en el caso inicial. Teniendo en cuenta que estamos en un dominio 2D, y que las distancias entre puntos es pequeña, lo que en realidad estamos haciendo con esta nueva expresión es promediar la derivada en un entorno alrededor del punto, disminuyendo así el posible ruido que podríamos haber obtenido en la discretización.

Para calcular la derivada parcial en la dirección y se procede exactamente igual, con lo que la expresión queda:

$$\begin{aligned} f_y(x_i, y_j) &= (1-\alpha)\left(\frac{f(x_i, y_{j+1}) - f(x_i, y_{j-1})}{2h}\right) + \\ &+\frac{\alpha}{2}\left(\frac{f(x_{i-1}, y_{j+1}) - f(x_{i-1}, y_{j-1}) + f(x_{i+1}, y_{j+1}) - f(x_{i+1}, y_{j-1})}{2h}\right) \end{aligned} \tag{1.2}$$

considerando que h también es la distancia entre valores vecinos de y.

1.1.4 Máscaras para el cálculo de las derivadas parciales

Con la discretización obtenida, el proceso para calcular las derivadas parciales es equivalente a convolucionar la imagen usando las máscaras siguientes:

$$\begin{aligned} H_x &= \frac{1}{2h}\begin{bmatrix} -\alpha/2 & 0 & \alpha/2 \\ -(1-\alpha) & 0 & (1-\alpha) \\ -\alpha/2 & 0 & \alpha/2 \end{bmatrix} \\ H_y &= \frac{1}{2h}\begin{bmatrix} -\alpha/2 & -(1-\alpha) & -\alpha/2 \\ 0 & 0 & 0 \\ \alpha/2 & (1-\alpha) & \alpha/2 \end{bmatrix} \end{aligned} \tag{1.3}$$

donde se ha supuesto que el eje x crece horizontalmente hacia la derecha de la imagen, y que el eje y crece hacia abajo.

Revisando en la bibliografía [SON98, PRA01] vemos que muchas de las máscaras usadas tradicionalmente caen dentro de este esquema. Una revisión más amplia puede consultarse en [HAR92].

Por ejemplo, Prewitt [PRE66] usa $\alpha = 2/3$, obteniendo la máscara

$$H_x = \frac{1}{6h}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \tag{1.4}$$

donde cada fila tiene igual importancia en el cálculo de la derivada.

Sobel [SOB70] usa $\alpha = 1/2$, lo cual produce la máscara

$$H_x = \frac{1}{8h}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

dando en este caso una importancia mayor (justo el doble) a la fila central donde se encuentra el pixel en cuestión que estamos estudiando.

Por último, Frei y Chen propusieron el caso $\alpha = 2 - \sqrt{2}$, quedando la máscara siguiente:

$$H_x = \frac{2-\sqrt{2}}{4h}\begin{bmatrix} -1 & 0 & 1 \\ -\sqrt{2} & 0 & \sqrt{2} \\ -1 & 0 & 1 \end{bmatrix} \tag{1.5}$$

Este último caso garantiza que, para bordes diagonales perfectos, la magnitud del gradiente resulte igual que en los bordes horizontales o verticales perfectos. Incidiremos en este caso particular en el capítulo siguiente.

Existen otras máscaras de distinto tamaño, como las de 2x2 propuesta por Roberts [ROB65], que tienen el problema de no estar centradas en el pixel que se está tratando, o de 5x5, como las propuestas en [JAI95], que exigen un costo computacional mucho mayor, aunque al tomar un entorno más amplio, son menos sensibles al ruido, y obtienen un resultado más suavizado (y por lo tanto a veces menos preciso). Pero la idea subyacente en todas ellas es la misma: obtener estimaciones para los valores de la derivada para, a partir de ahí, encontrar los bordes de la imagen.

1.1.5 Cálculo del vector gradiente

Una vez estimadas las derivadas parciales en un punto de la imagen, podemos estimar el vector gradiente en dicho punto. Sea $F(i,j)$ la imagen original, y sean $F_x(i,j)$ y $F_y(i,j)$ las dos imágenes resultantes de aplicar la convolución con las máscaras H_x y H_y respectivamente. El vector gradiente para cada pixel (i,j) será entonces

$$\nabla F(i,j) = [F_x(i,j), F_y(i,j)]$$

A partir de esta estimación, podemos obtener una imagen de gradientes (tomando únicamente el módulo) de la manera siguiente:

$$G(i,j) = \sqrt{F_x^2(i,j) + F_y^2(i,j)}$$

Con la información de dicha imagen podremos estimar que en aquellos puntos (i,j) donde el valor de $G(i,j)$ sea suficientemente grande, existirá un borde cuya orientación será perpendicular a la dirección del vector $[F_x(i,j), F_y(i,j)]$. En la figura 1.5 podemos ver un ejemplo de imagen de gradientes.

1.2 Hipótesis de trabajo

Acabamos de ver cómo el uso de una herramienta matemática como es el vector gradiente nos aporta bastante información sobre la existencia de bordes en una imagen. Sin embargo, para poder aplicar esta herramienta con rigor es preciso que nuestra función sea derivable (y por tanto continua) en todo su dominio (o al menos en aquellos puntos donde queramos evaluar el vector gradiente). Es por ello que todos estos trabajos toman como premisa fundamental el hecho de que, aunque nuestra imagen de entrada sea una señal con valores discretos, dicha señal proviene del muestreo de una cierta función continua y derivable, a la cual pretendemos calcular los bordes.

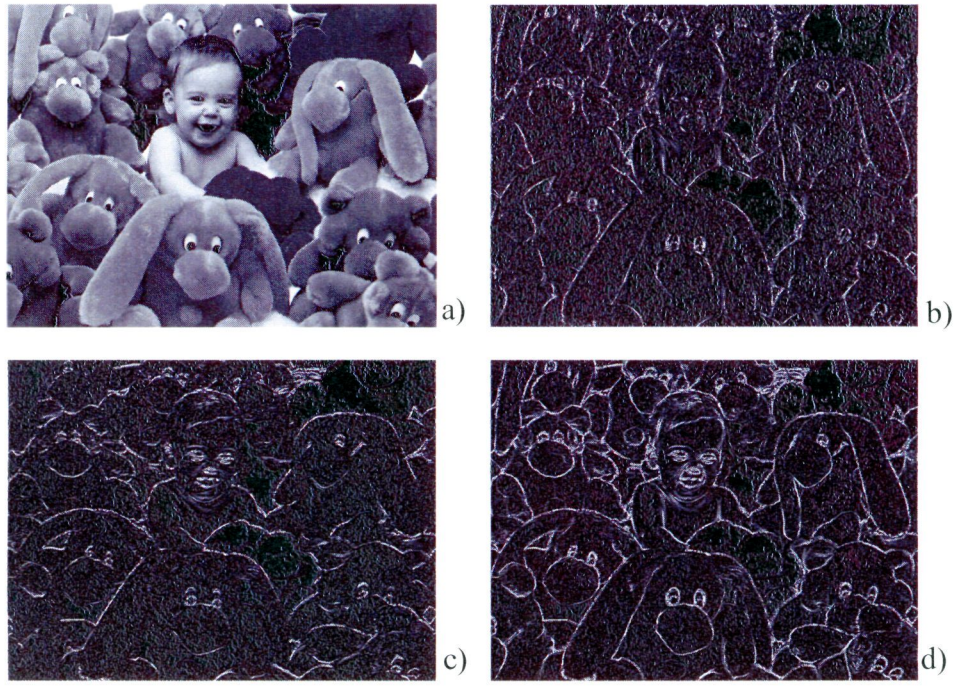


Figura 1.5: a) Imagen original. b) Parciales en x. c) Parciales en y. d) Módulo del gradiente

1.2.1 Modelo de borde

Gráficamente, y en un caso unidimensional, lo que estamos haciendo es asumir que en una imagen, antes de la digitalización, los bordes son como se ven en la figura 1.6a y no como se ven en la figura 1.6b, ya que en este último caso, habría una discontinuidad en el punto x_0 y por lo tanto no existiría ninguna derivada en ese punto. Para el caso bidimensional es lo mismo: se asume que la función cuya discretización poseemos en forma de imagen es continua en todo su dominio.

Sin embargo, a lo largo de este trabajo vamos a hacer una interpretación diferente. Vamos a asumir que justo en las zonas donde hay un borde en la imagen, **habrá una discontinuidad**. La razón es muy simple: supongamos la situación de la figura 1.7, donde la cámara[1] que adquiere la imagen está apuntando a dos objetos, cuyas intensidades son A y B. Justo en la zona de la imagen que delimita ambos objetos, existe una discontinuidad en la intensidad adquirida por la cámara.

Por lo tanto, para simular esta discontinuidad, nuestro **modelo de borde** va a ser el representado en la figura 1.6b, y será el que usaremos a lo largo de todo este trabajo. Usar dicho modelo va a implicar una serie de consecuencias. A partir de ahora, ya no tendrá sentido hablar del "gradiente en un pixel", sino del "vector normal al contorno que pasa por ese pixel". Y cuando, de ahora en adelante, hablemos de derivadas parciales en un pixel, estaremos refiriéndonos en realidad al resultado de aplicar lás máscaras de convolución anteriores, pero no al valor de la derivada en sí, puesto que damos por hecho que no existe.

Otro término que dejaremos de usar será "borde", y en su lugar usaremos "contorno". De esta manera, usaremos el término "borde" para indicar que un pixel es borde o no, y usaremos "contorno" para referirnos a la verdadera línea de grosor infinitesimal donde sucede la discontinuidad.

[1] Se ha supuesto que la cámara es de tipo "pin-hole", donde cada punto de la escena tridimensional se proyecta en un único punto sobre el plano donde se forma la imagen.

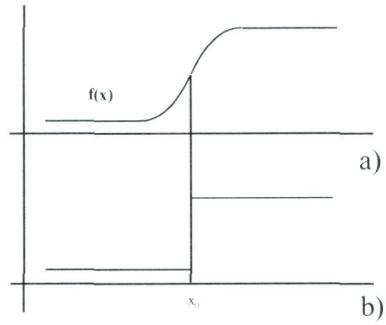


Figura 1.6: Representación unidimensional de un borde: a) real. b) ideal

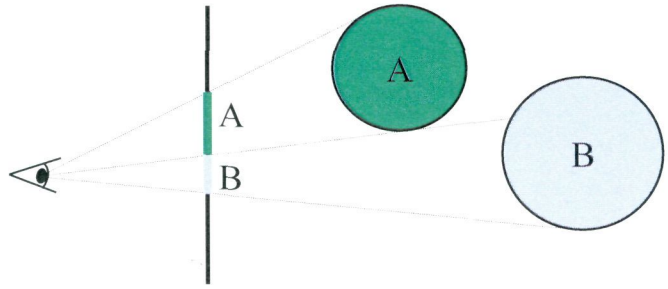


Figura 1.7: Discontinuidad producida en los bordes

1.2.2 Modelo de adquisición

La pregunta que surge ahora es relativa al muestreo de la imagen, es decir, a la adquisición de la fotografía: *si realmente existe una discontinuidad, ¿qué ocurre con los pixels donde cae justo esa discontinuidad?* ¿Qué intensidad tendrán? *¿A? ¿B?* Para responder a estas preguntas, usaremos el siguiente **modelo de adquisición**: la intensidad final del pixel será un promedio entre ambas intensidades, A y B, ponderados por el área relativa que cada valor de intensidad ocupa dentro del pixel, considerando éste como un pequeño cuadrado dentro de la imagen. Es decir, atendiendo a la figura 1.8, el valor de intensidad del pixel seleccionado vendrá dado por la expresión

$$F(i,j) = \frac{AS_A + BS_B}{h^2}$$

donde A y B son las intensidades de ambos objetos, S_A y S_B son las áreas de las regiones ocupadas por ambos valores de intensidad respectivamente en el interior del pixel, y h es la longitud de cada lado del pixel. Se cumple por tanto que el área del pixel es igual a $S_A + S_B = h^2$.

Dicho modelo puede ser considerado como la hipótesis de partida para todo nuestro trabajo. Un ejemplo de que dicha hipótesis parece cumplirse puede verse en la imagen digitalizada de la figura 1.9, donde se observa que el color de los pixels por donde pasa el contorno es en realidad un promediado entre los colores existentes a ambos lados del contorno (el color de la chaqueta y el del fondo). Otro ejemplo también ilustrativo puede

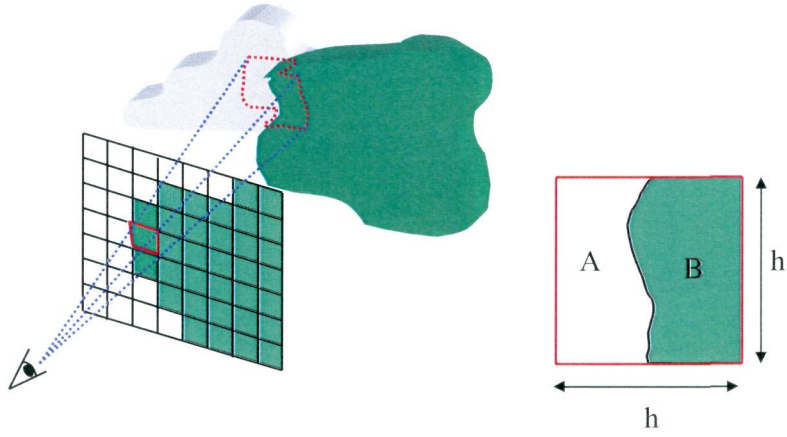


Figura 1.8: El color del pixel es un promedio entre A y B

verse en la figura 1.10, donde se ve la imagen de un objeto calibrador, usado para el calibrado de las cámaras[2]. Podemos ver cómo los pixels por donde pasa el contorno vertical adquieren valores del negro al blanco a medida que descendemos por el contorno, lo cual coincide con el comportamiento que nuestra hipótesis expone, puesto que la línea que forma dicho contorno no es vertical exactamente, sino que va desplazándose hacia la derecha de la imagen conforme va bajando. Esto hace que a medida que descendemos, el área interior al pixel que va quedando a la izquierda de la línea del contorno (correspondiente al color blanco) sea cada vez mayor, y por tanto el área a la derecha menor, lo que explica el aumento de la intensidad en el pixel.

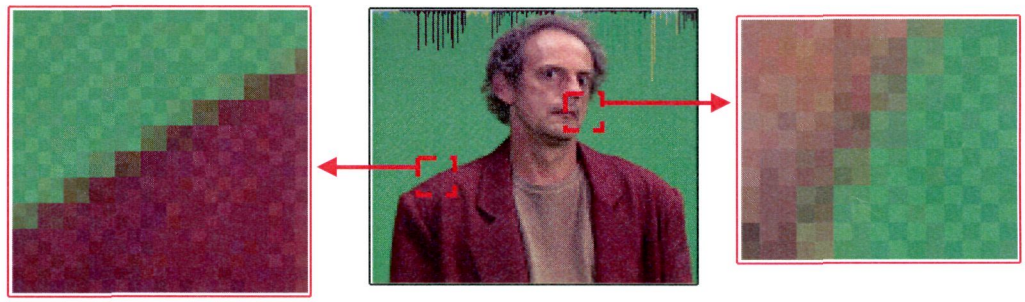


Figura 1.9: Los pixels por donde pasa el borde tienen un color intermedio entre ambos colores

La hipótesis planteada no es nueva. Ya existen otros autores que la mencionan, como se indica en [JAI89], pero nosotros vamos a analizar en profundidad todas las consecuencias que ello conlleva.

[2] Para calibrar una cámara suelen usarse imágenes donde puedan localizarse con precisión puntos concretos (como las esquinas de los rectángulos negros en la figura 1.10) cuya posición tridimensional en el espacio sea conocida a priori por el programa que realiza la calibración.

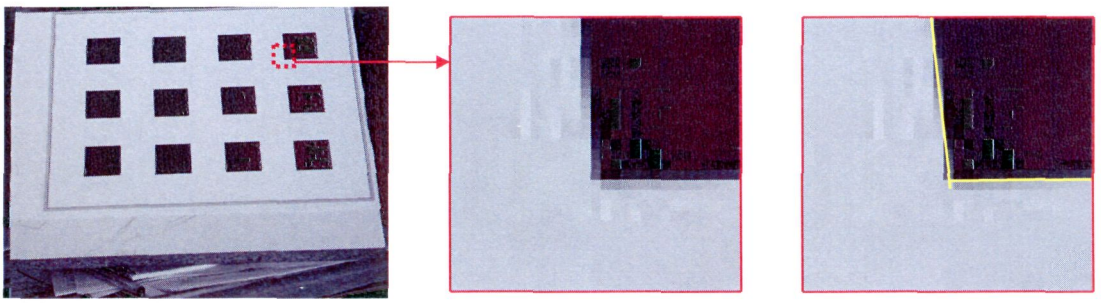


Figura 1.10: a) Imagen original. b) Imagen ampliada. c) Imagen ampliada con el contorno real superpuesto.

1.2.3 Interpretación del modelo de adquisición

La digitalización hecha por una cámara consiste en muestrear la intensidad proveniente de la escena sobre una malla rectangular de puntos, (x_i, y_j). En una cámara CCD, dicha malla se corresponde con una matriz de fotodiodos que captan la luz. Lo que estamos asumiendo con nuestro modelo es que dichos fotodiodos no recogen exactamente la iluminación en dichos puntos, sino en un cierto área alrededor de ellos (ver figura 1.11).

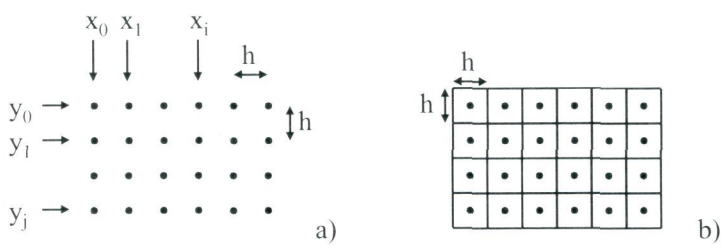


Figura 1.11: a) Matriz de fotodiodos de la cámara. b) Área de influencia de cada fotodiodo.

Si se asume que cada fotodiodo es uniforme e igualmente sensible a la luz, entonces la intensidad recogida en cada uno vendrá dada por la integral sobre el área abarcada por cada fotodiodo. Es decir,

$$F(x_i, y_j) = \frac{1}{h^2} \int_{y_j - h/2}^{y_j + h/2} \int_{x_i - h/2}^{x_i + h/2} f(x, y) dx dy \tag{1.6}$$

donde $f(x, y)$ es la intensidad que llega a la cámara, y $F(x_i, y_j)$ es la intensidad final adquirida en cada fotodiodo. Esta operación en realidad puede verse como la convolución con una función caja, $k(x, y)$, seguido de un muestreo en los puntos de la malla. Dicha función $k(x, y)$ estaría definida como

$$k(x, y) = \begin{cases} 1/h^2 & \text{si } |x|, |y| < h/2 \\ 0 & \text{en caso contrario} \end{cases}$$

Estos dos pasos pueden separarse. Por un lado podemos en primer lugar realizar la convolución continua de la imagen con la función $k(x, y)$, y posteriormente realizar el muestreo. Así podemos generalizar el proceso de formación de la imagen y separarlo del proceso de muestreo. La formación de la imagen vendría dada por la

expresión

$$g(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(u,v)k(x-u,y-v)dudv = f(x,y) * k(x,y)$$

El muestreo puede expresarse matemáticamente como la multiplicación de la señal continua por un tren de deltas, posicionado sobre los puntos donde se quiere realizar el muestreo. La expresión resultaría así:

$$F(x,y) = g(x,y)\sum_i\sum_j \delta(x-x_i, y-y_j)$$

Estos pasos pueden verse en la figura 1.12 para un caso unidimensional. La convolución transforma la discontinuidad en un salto suave de amplitud horizontal h, y posteriormente el muestreo produce que en el pixel donde se proyecta el borde se obtenga un valor de intensidad intermedio entre A y B.

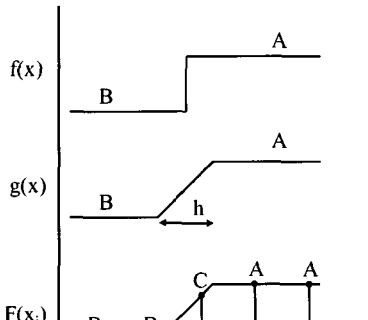


Figura 1.12: $f(x)$ representa la señal de entrada. $g(x)$ representa la señal recogida por los fotodiodos. $F(x_i)$ representa la señal discreta adquirida por la cámara.

1.3 Error cometido por las máscaras H_x y H_y

Ahora que tenemos una hipótesis sobre la intensidad que los pixels del borde deben tomar, podemos hacer un estudio de la exactitud de la estimación del vector gradiente usando las máscaras de convolución anteriores. Queremos ver si realmente el vector gradiente que obtenemos es normal al contorno, y si su módulo representa realmente la diferencia de intensidad a ambos lados del contorno. Todas estas máscaras ya han sido testeadas en numerosas ocasiones en casos ideales, utilizando imágenes sintéticas, en donde una imagen con un objeto oscuro de intensidad A sobre un fondo B estaba formado por pixels exclusivamente de uno de estos dos colores (fig.1.13) . Ahora que sabemos que a lo largo del borde podemos tener pixels con intensidades intermedias entre ambas, ¿cuál será el comportamiento de dichas máscaras? ¿Darán con exactitud el verdadero valor de módulo y dirección (en este caso por módulo esperamos obtener la magnitud $|A-B|$)? Estudiemos diferentes casos. Para ello vamos a tomar en nuestro estudio imágenes con un único contorno recto que atraviese toda la imagen, dejando las intensidades A y B $(A > B)$ a ambos lados del contorno.

1.3.1 Contorno horizontal sin desplazamiento

Sea F una imagen con un contorno horizontal sin desplazamiento dentro del pixel; esto es, el contorno cae exactamente en la frontera entre dos filas de pixels adyacentes, como se ve en la figura 1.14. Así tenemos que

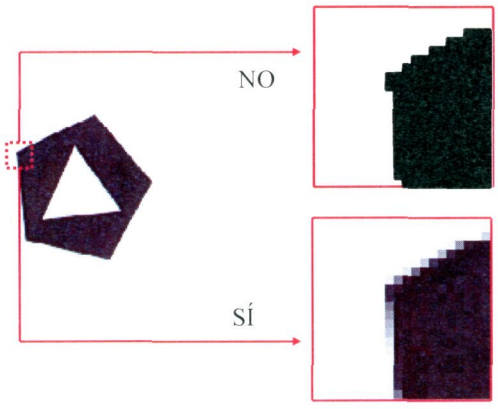


Figura 1.13: Un objeto negro sobre fondo blanco tiene pixels con colores intermedios en el borde

todos los pixels por encima del contorno tienen intensidad B, y todos los de abajo tienen intensidad A.

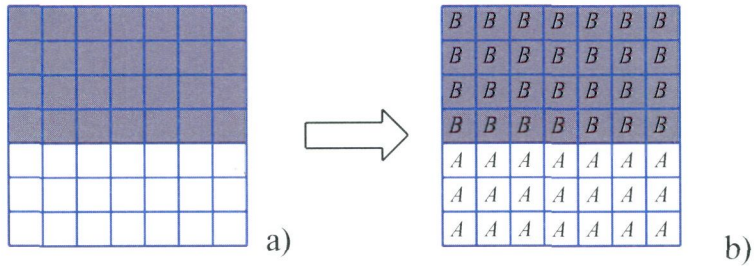


Figura 1.14: Contorno horizontal sin desplazamiento: a) Imagen original. b) Imagen digitalizada.

Si convolucionamos con la máscara H_y (ecuación 1.3) obtenemos la siguiente imagen[3]

$$F_y = \frac{1}{2h}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ A-B & A-B & A-B & A-B & A-B \\ A-B & A-B & A-B & A-B & A-B \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{1.7}$$

Convolucionar con la máscara H_x en este caso produciría una imagen con valor 0 en todos sus pixels. Por lo tanto, la imagen G que indica el módulo del gradiente sería exactamente igual a F_y. Lo primero que vemos es que hay dos filas con valor de gradiente alto, en lugar de una sola fila como sería deseable. En cualquier caso, vemos que el valor estimado para el módulo del gradiente es correcto (es igual al salto de intensidad a ambos lados del borde, dividido por el doble de la longitud del pixel), así como el de la orientación, ya que el vector tiene componente x nula, y por lo tanto apunta exactamente hacia abajo.

[3]La imagen de parciales que mostramos tiene menor resolución (5x5 en lugar de 7x7) porque se ha obviado el cálculo en los márgenes de la imagen original.

1.3.2 Contorno horizontal con desplazamiento

Sea F una imagen con un contorno horizontal cuya proyección caiga en el interior de la fila central de nuestra imagen, como se ve en la figura 1.15. Ahora tenemos que la fila central tendrá intensidad C, donde la expresión para C será, según nuestra hipótesis, la siguiente:

$$C = \frac{t}{h}A + \frac{h-t}{h}B$$

siendo t el desplazamiento del contorno dentro del pixel ($0 < t < h$).

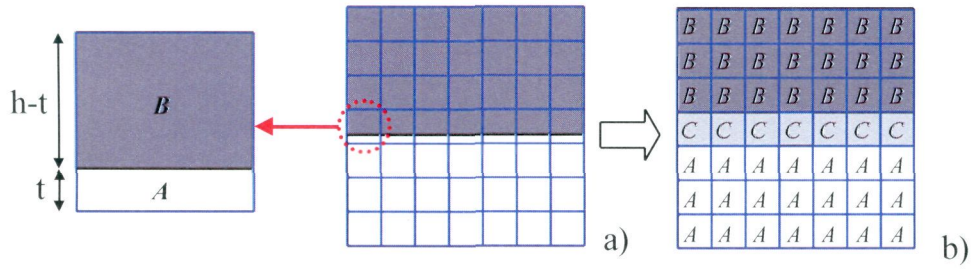


Figura 1.15: Contorno horizontal con desplazamiento: a) Imagen original. b) Imagen digitalizada.

Apliquemos ahora la convolución con H_y y obtenemos la siguiente imagen:

$$F_y = \frac{1}{2h}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ C-B & C-B & C-B & C-B & C-B \\ A-B & A-B & A-B & A-B & A-B \\ A-C & A-C & A-C & A-C & A-C \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{1.8}$$

De nuevo la convolución con H_x produce una imagen nula, y por lo tanto $G = F_y$. Curiosamente, ahora son tres las filas con valor no nulo del gradiente, aunque podemos ver que el valor de la fila central siempre será mayor que el de sus dos filas vecinas, ya que $B < C < A$. Si obviamos esas filas vecinas y nos quedamos con la central, cuyo valor es máximo, concluimos que para este caso la estimación del módulo y de la orientación del gradiente también es exacta.

1.3.3 Contorno de 45º sin desplazamiento

Sea F una imagen con un contorno diagonal de pendiente 1 que cruce los pixels exactamente de una esquina a otra, como se ve en la figura 1.16. Dicha imagen tendrá una diagonal de pixels cuya intensidad será

$$C = \frac{A+B}{2}$$

Al ser un contorno de 45 grados, la convolución con ambas máscaras, H_x y H_y producirá idénticos resultados. Sin embargo, los resultados serán diferentes en función del tipo de máscara que cojamos (Prewitt, Sobel o Frei-Chen). Hagámoslo entonces para un caso general, dejando α como parámetro:

$$F_x = F_y = \frac{1}{2h}\begin{bmatrix} 0 & 0 & P_{-2} & P_{-1} & P_0 \\ 0 & P_{-2} & P_{-1} & P_0 & P_1 \\ P_{-2} & P_{-1} & P_0 & P_1 & P_2 \\ P_{-1} & P_0 & P_1 & P_2 & 0 \\ P_0 & P_1 & P_2 & 0 & 0 \end{bmatrix}$$

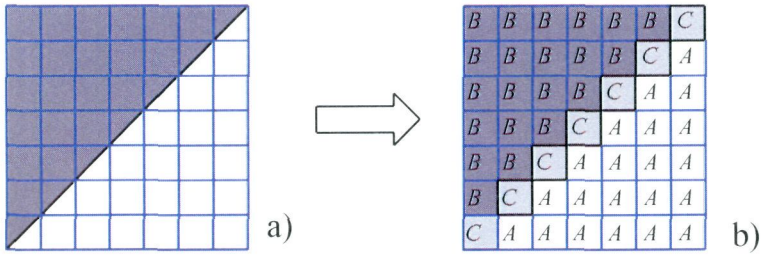


Figura 1.16: Contorno de 45° sin desplazamiento: a) Imagen original. b) Imagen digitalizada

donde

$$\begin{aligned} P_{-2} &= \frac{\alpha}{2}(C-B) \\ P_{-1} &= \frac{\alpha}{2}(A-B)+(1-\alpha)(C-B) \\ P_0 &= \frac{\alpha}{2}(A-B)+(1-\alpha)(A-B)=\left(\frac{2-\alpha}{2}\right)(A-B) \\ P_1 &= \frac{\alpha}{2}(A-B)+(1-\alpha)(A-C) \\ P_2 &= \frac{\alpha}{2}(A-C) \end{aligned}$$

Calculemos ahora la imagen de gradientes G. Teniendo en cuenta que ambas parciales son iguales, obtenemos el siguiente resultado:

$$G=\sqrt{2}F_x=\frac{1}{\sqrt{2}h}\begin{bmatrix} 0 & 0 & P_{-2} & P_{-1} & P_0 \\ 0 & P_{-2} & P_{-1} & P_0 & P_1 \\ P_{-2} & P_{-1} & P_0 & P_1 & P_2 \\ P_{-1} & P_0 & P_1 & P_2 & 0 \\ P_0 & P_1 & P_2 & 0 & 0 \end{bmatrix}$$

Vemos que el resultado es una imagen con una franja de 5 líneas diagonales cuyo perfil es creciente hasta la diagonal central y luego decreciente. Es decir, se cumple que

$$0<P_{-2}<P_{-1}<P_0>P_1>P_2>0$$

ya que, atendiendo a las expresiones de cada uno, todos los términos son positivos, y además se cumple que $B<C<A$. Esto significa que, de nuevo, a pesar de que alrededor del borde se produce una franja de grosor mayor que uno con valores no nulos en el gradiente, tomaremos como valor significativo el de la diagonal central, puesto que es allí donde se encuentra el valor más alto. Este valor es

$$\frac{1}{\sqrt{2}h}P_0=\frac{2-\alpha}{2\sqrt{2}h}(A-B) \tag{1.9}$$

Aquí ya vemos una diferencia con respecto a los casos anteriores con el borde vertical, y es que el valor del módulo del gradiente depende del parámetro α empleado en la máscara. Por lo tanto, sólo hay un valor de α que me garantice que el módulo sea el valor deseado, y es

$$\frac{2-\alpha}{2\sqrt{2}h}(A-B)=\frac{A-B}{2h}\Rightarrow\alpha=2-\sqrt{2}$$

que no es otro que el valor usado por Frei y Chen. Esto significa que las otras máscaras me producirán un error en el módulo del gradiente para este tipo de borde. Por otro lado, en cuanto a la dirección del vector gradiente, se obtiene de forma correcta para cualquier valor de α, puesto que al ser ambas parciales idénticas, el vector siempre es paralelo a la dirección $[1, 1]$.

1.3.4 Contorno de 45º con desplazamiento

Sea F una imagen con un contorno diagonal de pendiente 1 que cruce los pixels pero sin tocar las esquinas de éstos, como se ve en la figura 1.17. Dicha imagen tendrá dos diagonales de intensidades intermedias entre A y B. Estos valores de intensidad serán:

$$\begin{aligned} C &= \frac{\frac{1}{2}(h-t)^2}{h^2}B + \frac{h^2 - \frac{1}{2}(h-t)^2}{h^2}A = A - \frac{(h-t)^2}{2h^2}(A-B) \\ D &= \frac{\frac{1}{2}t^2}{h^2}A + \frac{h^2 - \frac{1}{2}t^2}{h^2}B = B + \frac{t^2}{2h^2}(A-B) \end{aligned}$$

siendo t el desplazamiento del contorno dentro del pixel $(0 < t < h)$. Vemos también que se cumple que $C > D$, $\forall t \in (0, h)$, ya que

$$C - D = (A-B)\left(\frac{h^2 + 2ht - 2t^2}{2h^2}\right) > 0$$

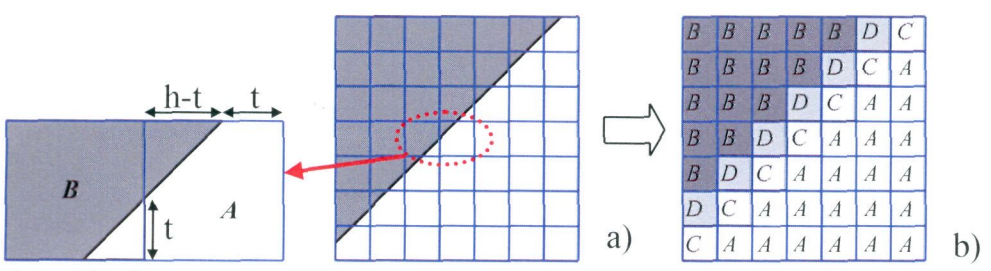


Figura 1.17: Contorno de 45º con desplazamiento: a) Imagen original. b) Imagen digitalizada

De nuevo, al ser un contorno de 45 grados, la convolución con ambas máscaras, H_x y H_y producirá idénticos resultados. Las imágenes resultantes serán[1]:

$$F_x = F_y = \frac{1}{2h}\begin{bmatrix} 0 & 0 & 0 & P_{-3} & P_{-2} & P_{-1} & P_0 \\ 0 & 0 & P_{-3} & P_{-2} & P_{-1} & P_0 & P_1 \\ 0 & P_{-3} & P_{-2} & P_{-1} & P_0 & P_1 & P_2 \\ P_{-3} & P_{-2} & P_{-1} & P_0 & P_1 & P_2 & 0 \\ P_{-2} & P_{-1} & P_0 & P_1 & P_2 & 0 & 0 \\ P_{-1} & P_0 & P_1 & P_2 & 0 & 0 & 0 \\ P_0 & P_1 & P_2 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{1.10}$$

donde

[1] Ahora vuelvo a mostrar imágenes de parciales 7x7 para que se aprecie mejor la franja de resultados no nulos que aparece.

$$
\begin{aligned}
P_{-3} &= \frac{\alpha}{2}(D-B) \\
P_{-2} &= \frac{\alpha}{2}(C-B)+(D-B)(1-\alpha) \\
P_{-1} &= \frac{\alpha}{2}(A-B)+(C-B)(1-\alpha) \\
P_{0} &= \frac{\alpha}{2}(A-B)+(A-D)(1-\alpha) \\
P_{1} &= \frac{\alpha}{2}(A-D)+(A-C)(1-\alpha) \\
P_{2} &= \frac{\alpha}{2}(A-C)
\end{aligned}
$$

Al igual que en el caso anterior, existe una franja diagonal de 6 líneas (antes eran 5) con el mismo perfil de antes: crece desde cero, alcanza un máximo y decrece de nuevo hasta cero. Es fácil comprobar que esto es cierto para la mayoría de los valores P_i, conociendo que $B < D < C < A$. El caso más difícil de ver se presenta entre P_{-1} y P_0, donde la cuestión radica en conocer si $A - D > C - B$. La respuesta es que depende del valor t, ya que

$$(A-D)-(C-B) = (A-B)\left(\frac{1}{2}-\frac{t}{h}\right)$$

Esto significa que $P_0 > P_{-1}$ si y sólo si $t < h/2$. Observando los pixels seleccionados en la figura 1.17, donde P_{-1} corresponde al pixel de la izquierda y P_0 al de la derecha, vemos que la condición es equivalente a decir que el tramo del contorno que pasa por el interior del pixel correspondiente a P_0 tiene **mayor longitud** que el tramo del contorno que pasa por el correspondiente a P_{-1}. En cierto sentido, parece lógico, ya que si tuviésemos que elegir entre los dos pixels cuál merece ser considerado como más perteneciente al borde que el otro, sería un buen criterio elegir al de P_0 por el argumento mencionado.

Cojamos pues este caso, $t < h/2$, con lo cual el máximo valor de la convolución se presenta en la diagonal central, y estudiemos cuál es el valor del módulo del gradiente en dicha franja. De nuevo, por igualdad entre ambas parciales, se cumple que

$$G = \sqrt{2}F_x = \frac{1}{\sqrt{2}h}\begin{bmatrix}
0 & 0 & 0 & P_{-3} & P_{-2} & P_{-1} & P_0 \\
0 & 0 & P_{-3} & P_{-2} & P_{-1} & P_0 & P_1 \\
0 & P_{-3} & P_{-2} & P_{-1} & P_0 & P_1 & P_2 \\
P_{-3} & P_{-2} & P_{-1} & P_0 & P_1 & P_2 & 0 \\
P_{-2} & P_{-1} & P_0 & P_1 & P_2 & 0 & 0 \\
P_{-1} & P_0 & P_1 & P_2 & 0 & 0 & 0 \\
P_0 & P_1 & P_2 & 0 & 0 & 0 & 0
\end{bmatrix}$$

con lo cual la expresión para el pixel central es

$$\frac{1}{\sqrt{2}h}P_0 = \frac{1}{\sqrt{2}h}(A-B)\left(\frac{2-\alpha}{2}-\frac{t^2}{2h^2}(1-\alpha)\right) = (A-B)\left(\frac{2-\alpha}{2\sqrt{2}h}-\frac{t^2}{2\sqrt{2}h^3}(1-\alpha)\right)$$

Esta expresión un tanto complicada se compone de dos términos. El primero es el mismo que obtuvimos en el caso anterior (ecuación 1.9) cuando el contorno pasaba por las esquinas de los pixels ($t = 0$). El segundo término vemos que depende también del parámetro α pero también depende de t, lo que significa que, sea cual sea la máscara que usemos para evaluar las parciales, el valor del módulo va a depender de la posición exacta del contorno dentro del pixel, lo cual no debería ocurrir. Por lo tanto deducimos que esta forma de evaluar el módulo del gradiente es errónea en un sentido riguroso, ya que no es independiente de la localización exacta

del contorno. Al menos, la dirección del vector sí sigue funcionando de forma exacta, pues de nuevo tenemos ambas parciales con idéntico valor. Sin embargo, veremos en el próximo y último caso, que también la dirección es errónea cuando la pendiente del contorno no es exactamente 45 grados.

1.3.5 Contorno de pendiente 1/2 sin desplazamiento

Sea F una imagen con un contorno diagonal de pendiente 1/2 que cruce exactamente entre la esquina inferior izquierda de un pixel y la esquina opuesta del pixel vecino de su derecha, tal y como se ve en la figura 1.18. Dicha imagen tendrá una diagonal en escalera (caminando dos pixels hacia la derecha en cada fila, y subiendo luego en diagonal), cuyos colores serán:

$$\begin{aligned} C &= \frac{3A+B}{4} \\ D &= \frac{A+3B}{4} \end{aligned}$$

donde se cumple que $B < D < C < A$.

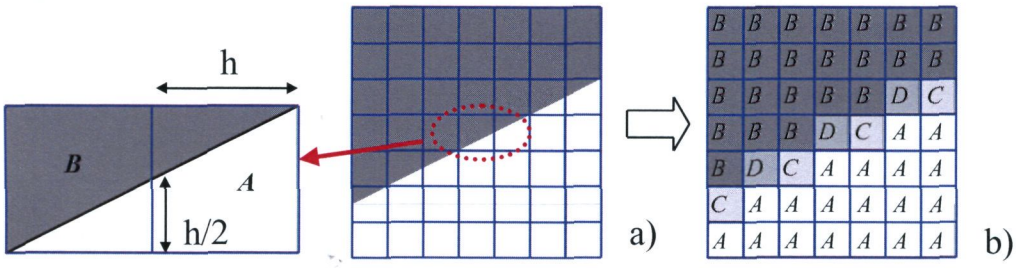


Figura 1.18: Contorno de pendiente 1/2 sin desplazamiento: a) Imagen original. b) Imagen digitalizada.

A diferencia de todos los casos estudiados anteriores, esta vez cada máscara va a producir una imagen diferente. La convolución con la máscara horizontal produce la imagen siguiente:

$$F_x = \frac{1}{2h}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & P_{-3} \\ 0 & 0 & 0 & 0 & P_{-3} & P_{-2} & P_{-1} \\ 0 & 0 & P_{-3} & P_{-2} & P_{-1} & P_0 & P_1 \\ P_{-3} & P_{-2} & P_{-1} & P_0 & P_1 & P_2 & P_3 \\ P_{-1} & P_0 & P_1 & P_2 & P_3 & P_4 & 0 \\ P_1 & P_2 & P_3 & P_4 & 0 & 0 & 0 \\ P_3 & P_4 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

donde

$$\begin{aligned} P_{-3} &= \frac{1}{2}\alpha\,(D-B) = \frac{1}{8}\alpha\,(A-B) \\ P_{-2} &= \frac{1}{2}\alpha\,(C-B) = \frac{3}{8}\alpha\,(A-B) \\ P_{-1} &= \frac{1}{2}\alpha\,(A-D) + (D-B)\,(1-\alpha) = \left(\frac{1}{4}+\frac{1}{8}\alpha\right)(A-B) \\ P_0 &= \frac{1}{2}\alpha\,(A-C) + (C-B)\,(1-\alpha) = \left(\frac{3}{4}-\frac{5}{8}\alpha\right)(A-B) \end{aligned}$$

$$
\begin{aligned}
P_1 &= \frac{1}{2}\alpha(D-B)+(A-D)(1-\alpha) = \left(\frac{3}{4}-\frac{5}{8}\alpha\right)(A-B)\\
P_2 &= \frac{1}{2}\alpha(C-B)+(A-C)(1-\alpha) = \left(\frac{1}{4}+\frac{1}{8}\alpha\right)(A-B)\\
P_3 &= \frac{1}{2}\alpha(A-D) = \frac{3}{8}\alpha(A-B)\\
P_4 &= \frac{1}{2}\alpha(A-C) = \frac{1}{8}\alpha(A-B)
\end{aligned}
$$

En esta ocasión la franja toma la dirección del borde y cruza la imagen con pendiente 1/2. Aparecen hasta 8 valores diferentes, y mirándola de izquierda a derecha mantiene el mismo perfil que en los casos anteriores: creciente, máximo y decreciente, ya que se cumple que:

$$0 < P_{-3} < P_{-2} < P_{-1} < P_0 = P_1 > P_2 > P_3 > P_4 > 0$$

donde además se cumple que $P_{-3} = P_4$, $P_{-2} = P_3$ y $P_{-1} = P_2$. Todas las inecuaciones pueden demostrarse fácilmente, excepto la que relaciona a P_{-1} y a P_0. Para que $P_{-1} < P_0$ debe cumplirse que:

$$P_0 - P_{-1} = \frac{1}{2} - \frac{3}{4}\alpha > 0 \Rightarrow \alpha < \frac{2}{3}$$

Curiosamente, este umbral para α es justo el que debe cumplirse para que el peso de la fila central sea mayor al peso de cada una de las filas vecinas en la máscara, tal y como se definieron en la ecuaciones 1.1 y 1.2. Es decir

$$1-\alpha \geqslant \frac{\alpha}{2} \Rightarrow \alpha \leqslant \frac{2}{3}$$

con lo cual se supone que se cumplirá en la mayoría de las máscaras. Un peso inferior en la fila central significaría que para estimar el valor de la derivada en una fila daríamos menos importancia a las intensidades de los pixels en esa fila que en las filas vecinas, lo cual sería un poco incoherente.

Veamos ahora la convolución con la máscara vertical. La imagen resultante es la siguiente:

$$
F_y = \frac{1}{2h}\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & Q_{-3}\\
0 & 0 & 0 & 0 & Q_{-3} & Q_{-2} & Q_{-1}\\
0 & 0 & Q_{-3} & Q_{-2} & Q_{-1} & Q_0 & Q_1\\
Q_{-3} & Q_{-2} & Q_{-1} & Q_0 & Q_1 & Q_2 & Q_3\\
Q_{-1} & Q_0 & Q_1 & Q_2 & Q_3 & Q_4 & 0\\
Q_1 & Q_2 & Q_3 & Q_4 & 0 & 0 & 0\\
Q_3 & Q_4 & 0 & 0 & 0 & 0 & 0
\end{bmatrix} \tag{1.11}
$$

donde

$$
\begin{aligned}
Q_{-3} &= \frac{1}{2}\alpha(D-B) = \frac{1}{8}\alpha(A-B)\\
Q_{-2} &= \frac{1}{2}\alpha(C-B)+(D-B)(1-\alpha) = \left(\frac{1}{4}+\frac{1}{8}\alpha\right)(A-B)\\
Q_{-1} &= \frac{1}{2}\alpha(A-2B+D)+(C-B)(1-\alpha) = \left(\frac{3}{4}-\frac{1}{8}\alpha\right)(A-B)\\
Q_0 &= \frac{1}{2}\alpha(A-2B+C)+(A-B)(1-\alpha) = \left(1-\frac{1}{8}\alpha\right)(A-B)
\end{aligned}
$$

$$
\begin{aligned}
Q_1 &= (A-B)(1-\alpha)+\frac{1}{2}\alpha(2A-B-D) = \left(1-\frac{1}{8}\alpha\right)(A-B) \\
Q_2 &= (A-D)(1-\alpha)+\frac{1}{2}\alpha(2A-B-C) = \left(\frac{3}{4}-\frac{1}{8}\alpha\right)(A-B) \\
Q_3 &= \frac{1}{2}\alpha(A-D)+(A-C)(1-\alpha) = \left(\frac{1}{4}+\frac{1}{8}\alpha\right)(A-B) \\
Q_4 &= \frac{1}{2}\alpha(A-C) = \frac{1}{8}\alpha(A-B)
\end{aligned}
$$

Vemos que también la imagen de parciales en y también presenta una franja similar con el mismo perfil, donde se cumple que

$$0 < Q_{-3} < Q_{-2} < Q_{-1} < Q_0 = Q_1 > Q_2 > Q_3 > Q_4 > 0$$

donde también se cumple que $Q_{-3} = Q_4$, $Q_{-2} = Q_3$ y $Q_{-1} = Q_2$.

Ahora habría que calcular la imagen del módulo del gradiente, que también tendrá un perfil similar al de las imágenes parciales, pero bastará con calcular el valor del gradiente en el pixel central, para demostrar que la estimación produce un resultado incorrecto. Justo en ese pixel, el vector gradiente será

$$\nabla F = \frac{1}{2h}[P_0, Q_0] = \frac{A-B}{2h}\left[\frac{3}{4}-\frac{5}{8}\alpha, 1-\frac{1}{8}\alpha\right]$$

Calculemos primero su módulo

$$|\nabla F| = \frac{A-B}{2h}\left(\frac{1}{4\sqrt{2}}\sqrt{13\alpha^2-38\alpha+50}\right) \neq \frac{A-B}{2h}$$

el cual vuelve a ser dependiente del valor del parámetro α, lo cual es incorrecto. Calculemos ahora su dirección. El vector gradiente tendrá una pendiente cuya arcotangente vendrá dada por el valor

$$\frac{Q_0}{P_0} = \frac{8-\alpha}{6-5\alpha} \neq 2$$

Ya que el vector gradiente debería ser normal al contorno, y como éste tiene pendiente 1/2 en este caso, la pendiente del vector debería ser 2 para cualquier valor de α, cosa que no ocurre.

1.4 Conclusiones

Tras el análisis realizado se observa que el esquema utilizado tradicionalmente para calcular el vector normal a los bordes de una imagen no es exacto en casos ideales, siempre partiendo de las hipótesis de trabajo (modelos de borde y de adquisición) que enunciamos al principio del capítulo. El vector gradiente, cuyo módulo debería representar la magnitud del salto de intensidad a ambos lados del borde, y cuya dirección debería ser ortogonal al contorno, toma valores diferentes según el valor que hayamos dado al parámetro α al evaluar las derivadas parciales. Además, aún con un valor fijo de α, el módulo puede variar según sea la localización exacta del contorno en el interior del pixel.

Por tanto, la pregunta que se plantea en este momento es la siguiente: ¿existirá algún esquema que nos dé con total exactitud el vector normal (módulo y dirección) para un contorno en cualquier orientación y posición dentro del pixel? En el próximo capítulo se desarrollará un nuevo esquema que responderá afirmativamente a esta cuestión. Más aún, dicho esquema no sólo encontrará el vector normal deseado, sino que además localizará la posición exacta dentro del pixel por donde pasa el contorno. Por supuesto, la exactitud total se conseguirá sólo en imágenes ideales (ideal en el sentido de las hipótesis planteadas).

Capítulo 2

Contornos de primer orden

- ¿Crees que existirá alguna manera de encontrar la orientación exacta?
- Podría intentarse.
- Si te das cuenta, toda la información necesaria está contenida en los valores de los pixels....

DICIEMBRE 2000

Acabamos de ver en el capítulo anterior que con el esquema tradicional de cálculo del gradiente, mediante el uso de máscaras de convolución para evaluar las derivadas parciales, los resultados no son exactos. A continuación trataremos de desarrollar una técnica que sí tenga resultados exactos, en el sentido que muestra la figura 2.1. Esto es, dada una foto donde haya un objeto sobre un fondo, obtener, para cada pixel por donde pase el borde de dicho objeto, el vector cuyo módulo represente el salto de intensidad entre el objeto y el fondo, y cuya orientación sea exactamente la dirección normal al contorno del objeto en ese punto. Hay que recordar que el término "contorno" lo usaremos para denominar la verdadera línea (con precisión real) donde se produce la discontinuidad desde el punto de vista de la cámara que ha adquirido la imagen, y no una línea discretizada sobre los pixels de la imagen.

Para ello, el objetivo es, a partir de los valores de intensidad de la imagen original, y conociendo que el valor de cada pixel ha sido obtenido según la hipótesis planteada en nuestro modelo de adquisición, tratar de obtener una expresión para la línea real que representa el contorno. Lógicamente, no podemos encontrar la expresión exacta, puesto que debido al proceso de adquisición, se ha perdido información del comportamiento del contorno en el interior del pixel. Por lo tanto, inicialmente, haremos la siguiente simplificación: supondremos que, localmente, el contorno sobre un pixel y su vecindad se comporta **como una línea recta**. Para ello, vamos a introducir las dos definiciones siguientes:

Definición de contorno de primer orden Llamaremos **contorno de primer orden** a la función intensidad siguiente

$$f(x,y,A,B,a,b,e) = \begin{cases} A & \text{si } ax+by+e \geq 0 \\ B & \text{si } ax+by+e < 0 \end{cases}$$

tal que a cada punto $(x,y) \in \mathbb{R}^2$ se le asocia una intensidad A o B en función de a qué lado de la recta $ax+by+e=0$ se encuentra, tal y como se ve en la figura 2.2a.

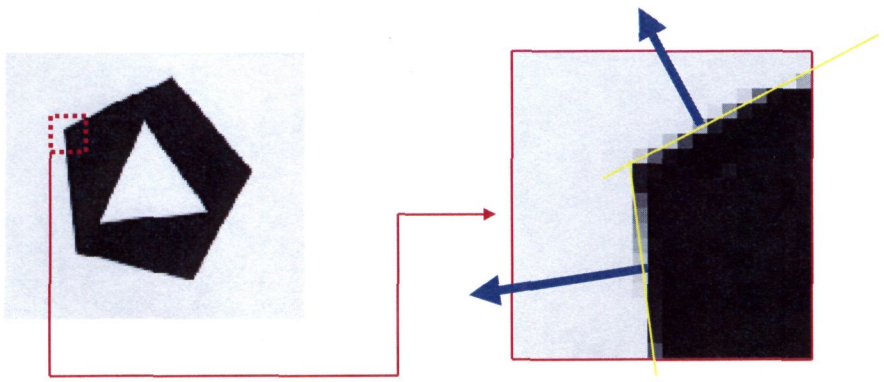


Figura 2.1: El objetivo inicial es localizar el vector normal al contorno en cada pixel, cuyo módulo represente el salto de intensidad a ambos lados del borde

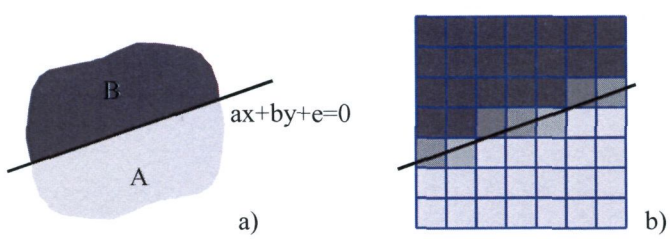


Figura 2.2: a) Contorno de primer orden, $f(x, y, A, B, a, b, e)$. b) Imagen ideal con un contorno de primer orden, $F_{i,j}(A, B, a, b, e)$

Por lo tanto, y haciendo uso de la expresión 1.6, llamaremos ahora **imagen ideal con un contorno de primer orden** a la imagen siguiente

$$F_{i,j}(A, B, a, b, e) = \int_{y_j - h/2}^{y_j + h/2} \int_{x_i - h/2}^{x_i + h/2} f(x, y, A, B, a, b, e) dx dy \tag{2.1}$$

donde $f(x, y, A, B, a, b, e)$ es un contorno de primer orden. Esta imagen puede verse en la figura 2.2b. Esto significa que el método que a continuación se desarrollará tratará de, partiendo sólamente de los valores de los pixels de la imagen $F_{i,j}$, encontrar los valores exactos para a, b, e y $A - B$.

La estructura del capítulo es la siguiente: al principio nos centraremos en el caso en que el contorno tiene una pendiente entre 0 y 1 (al que llamaremos primer octante), y para dicho caso obtendremos en primer lugar una expresión para obtener la diferencia de intensidad a ambos lados del contorno, luego otra para encontrar la orientación de dicho contorno, y finalmente una tercera expresión para encontrar la posición exacta del contorno dentro del pixel. Con esto tendríamos todos los parámetros del contorno obtenidos. A continuación generalizaremos la situación para el resto de octantes, y presentaremos el algoritmo para el caso general. Finalmente se verán algunos ejemplos de aplicar dicho algoritmo a varias imágenes sintéticas y reales.

2.1 Cálculo de la magnitud del borde (primer octante)

Comencemos por encontrar una forma de obtener el salto de intensidad producido a ambos lados del borde. Para ello, analicemos primero los resultados obtenidos con las derivadas parciales en el capítulo previo .

2.1.1 Derivadas parciales sobre un contorno

Atendiendo al estudio del capítulo previo sobre las imágenes resultantes de aplicar la convolución con las máscaras de cálculo de las derivadas parciales (sección 1.3), observamos que, ante la presencia de un borde, el resultado siempre es una franja de varias líneas de grosor, dependiendo de la pendiente del contorno. Así por ejemplo, para contornos horizontales, en la imagen F_y aparecían tres filas horizontales no nulas (ecuación 1.8), y para contornos de pendiente 1 podían aparecer hasta 6 líneas diagonales no nulas distintas (ecuación 1.10).

El perfil de todas ellas siempre era el mismo, como se ve en la figura 2.3. La imagen del módulo del gradiente es cero en las zonas homogéneas donde no hay borde, pero a medida que nos acercamos ortogonalmente a él, el módulo del gradiente crece hasta un máximo (justo en el pixel donde se encuentra el borde), y luego decrece hasta llegar a cero de nuevo a medida que atravesamos el borde y nos alejamos de él. Las imágenes de las parciales también siguen el mismo patrón.

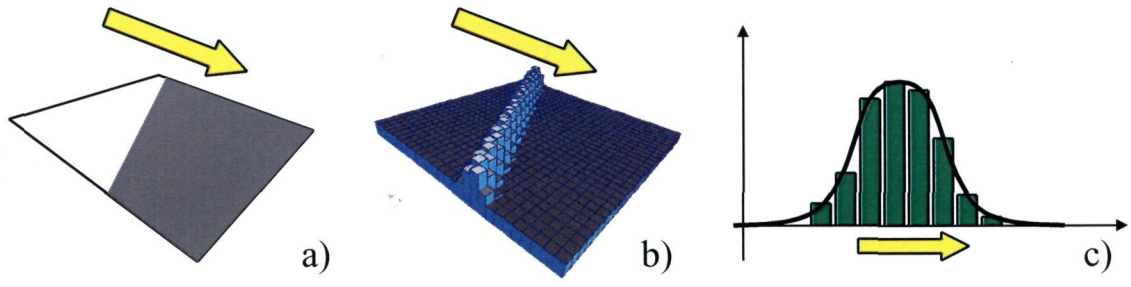


Figura 2.3: La imagen de gradientes presenta un perfil característico a lo largo de un contorno. a) Imagen con un contorno recto. b) Imagen del módulo del gradiente. c) Perfil de la imagen de gradiente en la dirección normal al contorno.

Cojamos en primer lugar el caso del contorno horizontal (sección 1.3.2), y tomemos la imagen de parciales en y (ecuación 1.8), ya que la de las parciales en x es una imagen nula. Vemos que aparecen tres filas con valores no nulos que forman la franja. Si cruzamos la franja de arriba a abajo en cualquiera de las columnas veremos el siguiente perfil

$$F_y = \frac{1}{2h} \begin{bmatrix} 0 \\ C-B \\ A-B \\ A-C \\ 0 \end{bmatrix}$$

Puede observarse que si sumamos los valores no nulos a lo largo de la columna, el resultado es siempre $(A-B)/h$, independientemente del valor de la posición t del contorno dentro del pixel (figura 1.15). Este resultado es igualmente válido para el caso del contorno horizontal sin desplazamiento (ecuación 1.7).

Miremos ahora el caso del contorno de 45 grados (sección 1.3.4), y tomemos cualquiera de las imágenes de parciales, ya que ambas son iguales (ecuación 1.10). Podríamos tratar de cruzar la franja de forma ortogonal

a la dirección del borde, pero al ser la imagen una matriz bidimensional de valores, no está claro cómo debería construirse la suma. Probemos entonces a realizar la suma en horizontal o en vertical (el resultado va a ser el mismo en ambos casos, ya que la imagen es simétrica con respecto a la diagonal). El perfil de la columna central tiene estos valores

$$F_y = \begin{bmatrix} 0 & P_{-3} & P_{-2} & P_{-1} & P_0 & P_1 & P_2 & 0 \end{bmatrix}$$

Si los sumamos todos obtenemos exactamente el mismo resultado, con independencia del desplazamiento t.

$$\sum_{i=-3}^{2} P_i = \frac{A-B}{h}$$

Cojamos por último el caso del contorno de pendiente 1/2 (sección 1.3.5), y tomemos de nuevo la imagen de parciales en y (ecuación 1.11), ya que presenta valores más significativos que en la de x [1]. Si sumamos cualquiera de sus columnas en vertical (hay dos tipos diferentes), obtenemos también idéntico resultado en ambas:

$$Q_{-3} + Q_{-1} + Q_1 + Q_3 = Q_{-2} + Q_0 + Q_2 + Q_4 = \frac{A-B}{h}$$

Lo interesante de esta expresión es que indica el salto exacto de intensidad que existe a ambos lados del contorno, dividido por la longitud del pixel. Por lo tanto, y a tenor de estos resultados, vamos a continuación a generalizar para ver si, para cualquier valor arbitrario de la pendiente, el resultado de la suma de los valores de las parciales a lo largo de una columna[2] sigue siendo el mismo.

2.1.2 Caso general para el primer octante

Sea $F_{i,j}(A, B, a, b, e)$ una imagen ideal con un contorno de primer orden (ecuación 2.1), donde el contorno atraviesa el pixel central de la imagen. Por simplicidad, dicho pixel tendrá coordenadas $(0, 0)$. Supondremos también que $A > B$. Por lo tanto, todo lo anterior significa que se cumple que

$$B \leq F_{0,0} \leq A$$

Como ya se ha indicado, tomemos además el caso en el que la pendiente del contorno está entre 0 y 1. Por lo tanto, el vector normal en este octante apuntará hacia abajo a la derecha, con mayor componente vertical que horizontal.

Si trazamos toda la familia de rectas posibles de pendiente entre 0 y 1 que pasen por el pixel central $(0, 0)$ obtenemos la región mostrada en color gris claro en la figura 2.4a. Esto significa que todos los pixels que no intersecten con dicha región tendrán valor A (si se encuentran por debajo del borde) o B (si se encuentran por encima), y que aquéllos que sí intersecten tendrán un valor V tal que $B \leq V \leq A$.

La imagen F por tanto tendrá la forma siguiente

$$F(i,j) = \begin{bmatrix} B & B & B & B & B & F_{2,-3} & F_{3,-3} \\ B & B & B & B & F_{1,-2} & F_{2,-2} & F_{3,-2} \\ B & B & B & F_{0,-1} & F_{1,-1} & F_{2,-1} & F_{3,-1} \\ F_{-3,0} & F_{-2,0} & F_{-1,0} & F_{0,0} & F_{1,0} & F_{2,0} & F_{3,0} \\ F_{-3,1} & F_{-2,1} & F_{-1,1} & F_{0,1} & A & A & A \\ F_{-3,2} & F_{-2,2} & F_{-1,2} & A & A & A & A \\ F_{-3,3} & F_{-2,3} & A & A & A & A & A \end{bmatrix} \tag{2.2}$$

[1] Al ser el vector gradiente ortogonal al borde, y tener este pendiente entre 0 y 1, su componente y (F_y) ha de ser mayor que su componente x (F_x).

[2] Hablamos de una columna porque estamos en el primer octante (pendiente del contorno entre 0 y 1). Posteriormente cuando se generalice al resto de octantes, veremos que para contornos con pendiente mayor que 1, la suma se hará a lo largo de la fila.

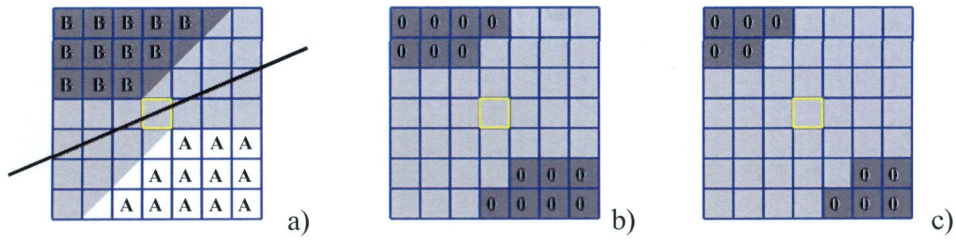


Figura 2.4: Contorno con pendiente entre 0 y 1. a) Imagen original. b) Imagen de parciales en y para el caso $\alpha = 0$. c) Imagen de parciales en y para el caso $\alpha > 0$.

donde el primer subíndice i indica la coordenada x del pixel, la cual crece de izquierda a derecha, y el segundo j indica la coordenada y del pixel, creciendo de arriba hacia abajo. Los valores indicados como $F_{i,j}$ son desconocidos a priori. Veamos ahora cómo es la expresión para las imágenes de las parciales, separando el caso cuando el parámetro α de las máscaras de convolución toma un valor nulo de cuando no.

Caso con $\alpha = 0$

Al tener la pendiente un valor entre 0 y 1, se cumple que los valores de las parciales en y son mayores que los de las parciales en x. Cojamos por tanto la imagen de parciales en y, f_y, a partir de una máscara h_y con $\alpha = 0$, esto es, usando simplemente los valores de la columna del pixel sobre el que se realiza la convolución[3]. Esta máscara tendrá la siguiente expresión

$$h_y = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Con esta máscara, el valor para cada pixel de la imagen f_y viene dado por la expresión siguiente:

$$f_y(i,j) = \frac{1}{2h}\left(F_{i,j+1} - F_{i,j-1}\right)$$

En la figura 2.4b se muestran en color gris claro todos aquellos pixels cuya parcial en y pudiera tener un valor no nulo (dependiendo de la posición exacta del contorno). Vemos que en la columna central aparecen 5 valores que pueden ser distintos de cero. Si sumamos todos ellos obtenemos

$$s_y = \sum_{j=-2}^{2} f_y(0,j) = \frac{A-B}{h}$$

que es el valor deseado para el módulo del vector que estamos buscando.

Caso con $\alpha > 0$

Si tomamos un valor de α no nulo estaremos usando la siguiente máscara

$$H_y = \frac{1}{2h}\begin{bmatrix} -\alpha/2 & -(1-\alpha) & -\alpha/2 \\ 0 & 0 & 0 \\ \alpha/2 & 1-\alpha & \alpha/2 \end{bmatrix}$$

[3] Usaremos letras minúsculas en el nombre de la imagen de parciales, f_y, para representar el caso con $\alpha = 0$.

La expresión para cada pixel de la imagen de parciales es ahora la siguiente:

$$F_y(i,j) = (1-\alpha)\left(\frac{F_{i,j+1} - F_{i,j-1}}{2h}\right) + \frac{\alpha}{2}\left(\frac{F_{i-1,j+1} - F_{i-1,j-1} + F_{i+1,j+1} - F_{i+1,j-1}}{2h}\right)$$

La imagen resultante tendrá entonces una zona mayor de valores que pueden ser no nulos (pixels con color gris claro en la figura 2.4c), ya que para cada valor de la imagen de parciales se toman en consideración un mayor número de pixels de la imagen original. En este caso, la columna central pasa a tener hasta 7 valores que pueden ser no nulos. La suma de todos ellos es también

$$S_y = \sum_{j=-3}^{3} F_y(0,j) = \frac{A-B}{h}$$

que es el mismo valor que antes. Con esto queda demostrado el siguiente lema:

Lema 2.1 *Sea $F_{i,j}(A,B,a,b,e)$ una imagen ideal con un contorno de primer orden que atraviesa el pixel central con pendiente entre 0 y 1. La suma de los 5 valores no nulos (7 en el caso que $\alpha > 0$) de la columna central de la imagen de parciales en y es igual a*

$$s_y = S_y = \frac{A-B}{h}$$

donde h es la longitud de cada pixel.

Diferencias entre usar α nulo o no

Realmente lo que está ocurriendo es sencillo de explicar. En el caso en que α es nulo se cumple que

$$s_y = \sum_{j=-2}^{2} f_y(0,j) = \frac{1}{2h}\left[(F_{0,2} + F_{0,3}) - (F_{0,-2} + F_{0,-3})\right] = \frac{A-B}{h}$$

Es decir, el resultado de sumar la columna de parciales en y es equivalente a estimar el valor de intensidad A a partir de los pixels $(0,2)$ y $(0,3)$, estimar B a partir de los pixels $(0,-2)$ y $(0,-3)$, y realizar la diferencia, tal y como se ve en la figura 2.5a.

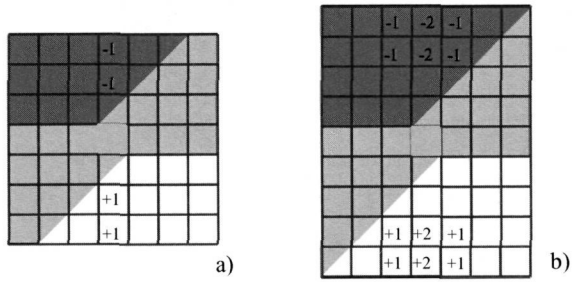


Figura 2.5: Pixels de la imagen original incluidos en la sumatoria de la columna de parciales en y: a) para $\alpha = 0$, b) para $\alpha = 1/2$. El número de cada pixel indica el peso que tiene dentro de la sumatoria.

En el caso α no nulo, la suma abarca más pixels, y el resultado se ve en la figura 2.5b. En este caso, para la estimación de los valores A y B se usa un conjunto mayor de pixels, dando más importancia a los de la columna central. Esto significa que el resultado será menos sensible al posible ruido que pueda existir en los valores de la imagen original.

2.2 Cálculo de la orientación del contorno (primer octante)

Para el cálculo de la magnitud del borde hemos usado exclusivamente la imagen de parciales en y, sin utilizar la imagen de parciales en x. Sin embargo, para el cálculo de la orientación del borde, ésta va a jugar un papel fundamental. Sólo hay que ver que, tal y como se muestra en la figura 2.6, el valor de la parcial en x es proporcional a la pendiente del borde. Por ejemplo, para bordes cercanos a 45^o, F_x es prácticamente igual que F_y. A medida que la pendiente decrece, también lo hará el valor de F_x. Cuando la pendiente es casi cero, también tiende a anularse el valor de F_x.

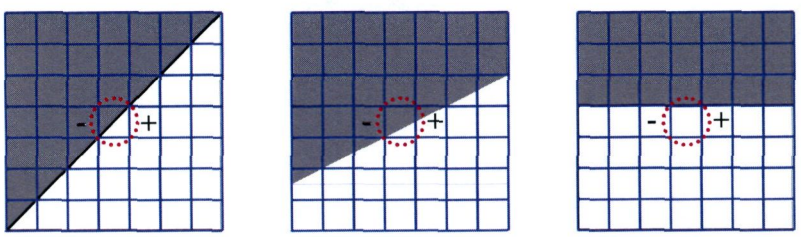


Figura 2.6: El valor de la parcial en x decrece conforme lo hace la pendiente del contorno

Si hacemos el cálculo de la sección anterior, sumando los valores no nulos de la columna central, pero usando los valores de la imagen de parciales en x, obtendremos un resultado a partir del cual se podrá estimar el valor de la pendiente. Separemos de nuevo en dos casos, según si α toma un valor nulo o no.

2.2.1 Caso general para el primer octante

Caso con $\alpha = 0$

Sea $F_{i,j}(A, B, a, b, e)$ la misma imagen de la ecuación 2.2, y cojamos ahora la imagen de parciales en x, f_x, a partir de una máscara h_x con $\alpha = 0$, esto es, usando simplemente los valores de la fila del pixel sobre el que se realiza la convolución. Esta máscara tendrá la siguiente expresión

$$h_x = \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

Cada pixel de la imagen f_x tendrá la expresión siguiente:

$$f_x(i,j) = \frac{1}{2h}\left(F_{i+1,j} - F_{i-1,j}\right)$$

La imagen f_x resultante aparece en la figura 2.7b, donde de nuevo los pixels con color gris claro indican que su parcial en x pudiera tener un valor no nulo. Vemos que en la columna central de nuevo aparecen 5 valores que pueden ser distintos de cero. Si sumamos todos ellos obtenemos:

$$s_x = \sum_{j=-2}^{2} f_x(0,j) = \frac{A-B}{h} + \frac{1}{2h}\left(\sum_{j=-2}^{0} F_{1,j} - \sum_{j=0}^{2} F_{-1,j}\right)$$

Más adelante analizaremos el significado de esta expresión. Vamos ahora con el caso $\alpha > 0$.

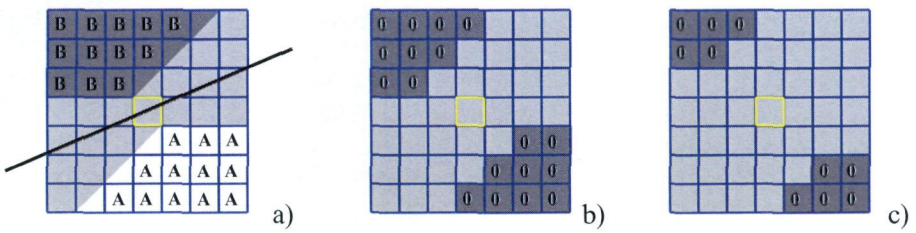


Figura 2.7: Contorno con pendiente entre 0 y 1. a) Imagen original. b) Imagen de parciales en x para el caso $\alpha = 0$. c) Imagen de parciales en x para el caso $\alpha > 0$.

Caso con $\alpha > 0$

Si tomamos un valor de α no nulo estaremos usando la siguiente máscara

$$H_x = \frac{1}{2h} \begin{bmatrix} -\alpha/2 & 0 & \alpha/2 \\ -(1-\alpha) & 0 & 1-\alpha \\ -\alpha/2 & 0 & \alpha/2 \end{bmatrix}$$

La expresión para cada pixel de la imagen de parciales es ahora la siguiente:

$$F_x(i,j) = (1-\alpha)\left(\frac{F_{i+1,j} - F_{i-1,j}}{2h}\right) + \frac{\alpha}{2}\left(\frac{F_{i+1,j-1} - F_{i-1,j-1} + F_{i+1,j+1} - F_{i-1,j+1}}{2h}\right)$$

La imagen F_x resultante tendrá entonces una zona mayor de valores que pueden ser no nulos, tal y como se ve en la figura 2.7c. En este caso, la columna central pasa a tener hasta 7 valores que pueden ser no nulos. La suma de todos ellos nos va a dar la misma expresión que el caso anterior

$$S_x = \sum_{j=-3}^{3} F_x(0,j) = \frac{A-B}{h} + \frac{1}{2h}\left(\sum_{j=-2}^{0} F_{1,j} - \sum_{j=0}^{2} F_{-1,j}\right) \tag{2.3}$$

Análisis de la expresión resultante

Visualmente, la expresión es una suma ponderada de los pixels que se muestran en la figura 2.8. Algunos de esos pixels sabemos de antemano que tendrán valor A o B, pero otros (mostrados en color gris) dependerán de la pendiente y la posición exacta del contorno. Por ejemplo, si el contorno fuera la recta r_1, los únicos pixels que tendrían valor intermedio entre A y B serían los de la fila central, ya que son los únicos pixels que toca la recta. El resto tendría valor A o valor B. Sin embargo, para la recta r_2, habrían más pixels con valores intermedios, pues la recta atraviesa un número mayor de pixels que antes.

Realmente, toda la familia de rectas de pendiente entre 0 y 1 que atraviesan el pixel central forman 12 diferentes configuraciones, atendiendo a los pixels que toca la recta, y a la altura a la que cruza de una columna a otra. Estos 12 casos se muestran en la figura 2.9.

Sea t la diferencia de altura entre la recta y la esquina inferior izquierda del pixel central (fig. 2.10). Y sea c la altura que sube la recta cuando nos desplazamos h unidades en horizontal. Es decir, la pendiente de la recta será c/h. Entonces, los 12 casos son los siguientes:

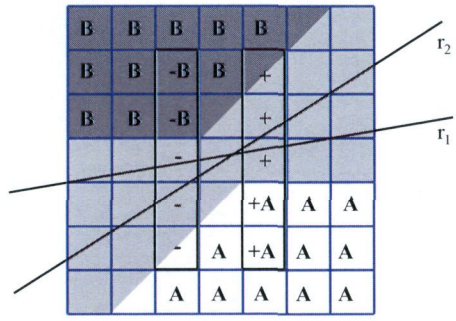


Figura 2.8: Pixels utilizados en la expresión para S_x.

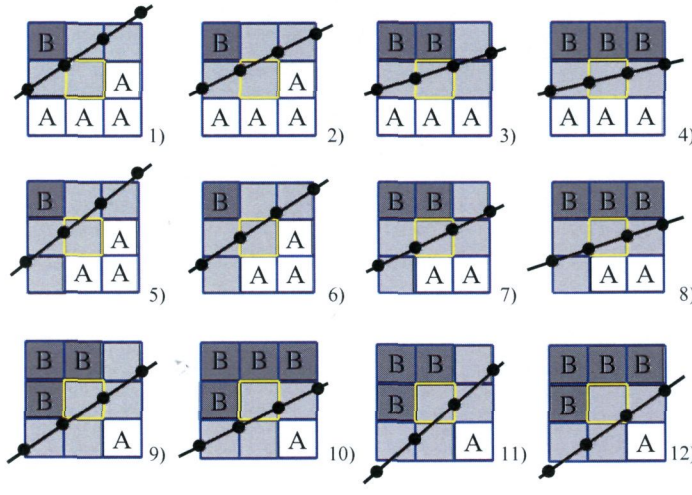


Figura 2.9: Todos los casos posibles en los que una recta de pendiente entre 0 y 1 pasa por el pixel central. Los puntos negros muestran por dónde cruza la recta entre las columnas de pixels.

1) $0 \leq t-c<h \quad 0 \leq t<h \quad h \leq t+c<2h \quad 2h \leq t+2c<3h$
2) $0 \leq t-c<h \quad 0 \leq t<h \quad h \leq t+c<2h \quad h \leq t+2c<2h$
3) $0 \leq t-c<h \quad 0 \leq t<h \quad 0 \leq t+c<h \quad h \leq t+2c<2h$
4) $0 \leq t-c<h \quad 0 \leq t<h \quad 0 \leq t+c<h \quad 0 \leq t+2c<h$
5) $-h \leq t-c<0 \quad 0 \leq t<h \quad h \leq t+c<2h \quad 2h \leq t+2c<3h$
6) $-h \leq t-c<0 \quad 0 \leq t<h \quad h \leq t+c<2h \quad h \leq t+2c<2h$
7) $-h \leq t-c<0 \quad 0 \leq t<h \quad 0 \leq t+c<h \quad h \leq t+2c<2h$
8) $-h \leq t-c<0 \quad 0 \leq t<h \quad 0 \leq t+c<h \quad 0 \leq t+2c<h$
9) $-h \leq t-c<0 \quad -h \leq t<0 \quad 0 \leq t+c<h \quad h \leq t+2c<2h$
10) $-h \leq t-c<0 \quad -h \leq t<0 \quad 0 \leq t+c<h \quad 0 \leq t+2c<h$
11) $-2h \leq t-c<-h \quad -h \leq t<0 \quad 0 \leq t+c<h \quad h \leq t+2c<2h$
12) $-2h \leq t-c<-h \quad -h \leq t<0 \quad 0 \leq t+c<h \quad 0 \leq t+2c<h$

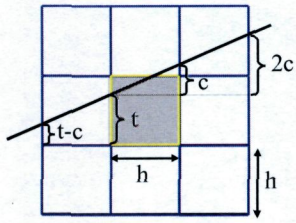


Figura 2.10:

Tomemos por ejemplo el caso 4, que aparece en la figura 2.10 con más detalle. Aplicando la expresión resultante para S_x (ecuación 2.3) obtenemos lo siguiente:

$$\begin{aligned} s_x &= S_x = \frac{A-B}{h} + \frac{1}{2h}\left(\sum_{j=-2}^{0} F_{1,j} - \sum_{j=0}^{2} F_{-1,j}\right) = \\ &= \frac{A-B}{h} + \frac{1}{2h}\left(B + F_{1,-1} + A - F_{-1,0} - 2A\right) = \\ &= \frac{A-B}{2h} + \frac{1}{2h}\left(F_{1,-1} - F_{-1,0}\right) \end{aligned}$$

Dicha expresión depende del valor de intensidad de los pixels $(1,-1)$ y $(-1,0)$. Dichos valores son

$$\begin{aligned} F_{1,-1} &= \frac{P_{1,-1}}{h^2}A + \frac{h^2 - P_{1,-1}}{h^2}B \\ F_{-1,0} &= \frac{P_{-1,0}}{h^2}A + \frac{h^2 - P_{-1,0}}{h^2}B \end{aligned}$$

donde $P_{1,-1}$ y $P_{-1,0}$ son las áreas interiores a dichos pixels que caen bajo la recta, y cuyas expresiones son:

$$\begin{aligned} P_{1,-1} &= (t + c - h)h + \frac{1}{2}ch \\ P_{-1,0} &= (t - c)h + \frac{1}{2}ch \end{aligned}$$

Sustituyendo en la expresión anterior para S_x obtenemos lo siguiente

$$s_x = S_x = \frac{A-B}{2h} + \frac{A-B}{2h^3}\left(P_{1,-1} - P_{-1,0}\right) = \frac{c}{h^2}\left(A - B\right)$$

Este resultado es muy interesante, ya que si lo dividimos por el valor obtenido para la suma de las parciales en y nos da exactamente la pendiente del contorno, que es el valor que queremos estimar:

$$\frac{s_x}{s_y} = \frac{S_x}{S_y} = \frac{c}{h}$$

Estudio del resto de casos Faltaría ver que en los 11 casos restantes se cumple esta misma propiedad. Eso implicaría calcular en cada uno las expresiones para el valor de intensidad de todos los pixels por donde pasa la recta. Afortunadamente, hay una manera de demostrar la propiedad para todos los casos a la vez, en lugar de

ir analizando caso a caso. Llamemos como hicimos antes $P_{i,j}$ al área interior al pixel (i,j) que cae bajo la recta. En el caso que la recta no toque el pixel en cuestión, $P_{i,j}$ valdrá 0 si la recta pasa por debajo del pixel, y h^2 si pasa por arriba. De esta forma tendremos que la intensidad de un pixel (i,j) vendrá dada por la expresión

$$F(i,j) = \frac{P_{i,j}}{h^2}A + \frac{h^2 - P_{i,j}}{h^2}B = B + \frac{(A-B)}{h^2}P_{i,j}$$

Esto significa que las dos sumatorias de la expresión de S_x (ecuación 2.3) valen

$$\begin{aligned}\sum_{j=-2}^{0} F_{1,j} &= 3B + \frac{(A-B)}{h^2}\sum_{j=-2}^{0} P_{1,j} \\ \sum_{j=0}^{2} F_{-1,j} &= 3B + \frac{(A-B)}{h^2}\sum_{j=0}^{2} P_{-1,j}\end{aligned}$$

con lo cual vemos que lo importante no es el valor de los $P_{i,j}$ individuales, sino el valor de la suma de ellos en una misma columna (para $i = 1$ ó -1), y esta suma tendrá siempre la misma expresión, independientemente de en cuál de los 12 casos estemos. Por ejemplo, atendiendo a la figura 2.11, para la columna de la derecha ($i = 1$) esta suma representa el área bajo la recta interior al rectángulo formado por los pixels $(1,0)$, $(1,-1)$ y $(1,-2)$, cuya expresión es

$$\sum_{j=-2}^{0} P_{1,j} = (t+c)\,h + \frac{1}{2}ch = th + \frac{3}{2}ch$$

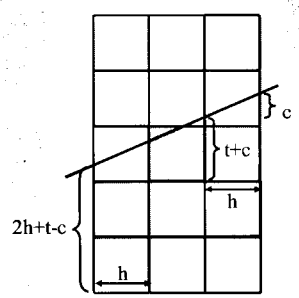


Figura 2.11:

Para la columna izquierda procedemos igual, obteniendo la expresión

$$\sum_{j=0}^{2} P_{-1,j} = (2h + t - c)\,h + \frac{1}{2}ch = 2h^2 + th - \frac{1}{2}ch$$

Por lo tanto, la expresión de la suma en todos los casos es equivalente a

$$s_x = S_x = \frac{A-B}{h} + \frac{1}{2h}\left(\sum_{j=-2}^{0} F_{1,j} - \sum_{j=0}^{2} F_{-1,j}\right) =$$

$$
\begin{aligned}
&= \frac{A-B}{h} + \frac{(A-B)}{2h^3}\left(\sum_{j=-2}^{0} P_{1,j} - \sum_{j=0}^{2} P_{-1,j}\right) = \\
&= \frac{A-B}{h} + \frac{(A-B)}{2h^3}\left(2ch - 2h^2\right) = \\
&= \frac{c}{h^2}(A-B)
\end{aligned}
$$

con lo cual, queda demostrado el siguiente lema:

Lema 2.2 *Sea $F_{i,j}(A, B, a, b, e)$ una imagen ideal con un contorno de primer orden que atraviesa el pixel central con pendiente entre 0 y 1. La suma de los 5 valores no nulos (7 en el caso que $\alpha > 0$) de la columna central de la imagen de parciales en x es igual a*

$$s_x = S_x = \frac{c}{h^2}(A-B)$$

donde h es la longitud de cada pixel, y c/h es la pendiente de dicho contorno.

Diferencias entre usar α nulo o no

La diferencia entre usar un valor de α nulo o no puede verse en la figura 2.12. Con α nulo se estarían tomando de la imagen original dos columnas de 5 pixels de altura para estimar el valor de la pendiente a partir de la resta de ambas. En el caso $\alpha > 0$, las columnas serían de 9 pixels de altura, aunque dando mayor peso a los 5 valores centrales, y un peso muy pequeño a los pixels más alejados.

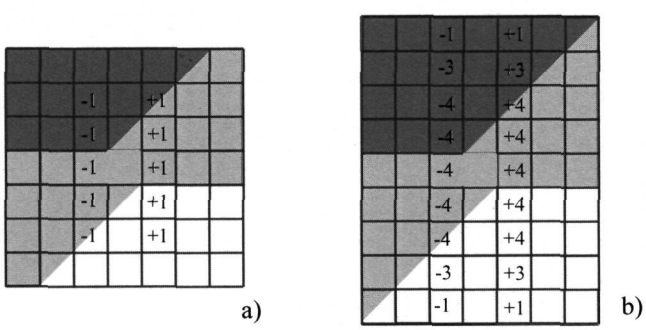


Figura 2.12: Pixels de la imagen original incluidos en la sumatoria de la columna de parciales en x: a) para $\alpha = 0$, b) para $\alpha = 1/2$. El número de cada pixel indica el peso que tiene dentro de la sumatoria.

2.2.2 Cálculo del vector normal al contorno (primer octante)

Ya estamos en condiciones de obtener el vector normal al contorno. Sea $N = [n_x, n_y]$ dicho vector (figura 2.13). Las condiciones que queremos que cumpla son dos

$$
\begin{aligned}
\|N\| &= \frac{A-B}{h} = S_y \\
\frac{n_y}{n_x} &= \frac{h}{c} = \frac{S_y}{S_x}
\end{aligned}
$$

Por lo tanto, de aquí ya pasamos a presentar el siguiente lema:

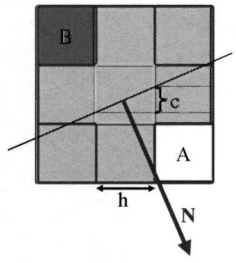


Figura 2.13: Vector normal al contorno

Lema 2.3 *Sea $F_{i,j}(A, B, a, b, e)$ una imagen ideal con un contorno de primer orden que atraviesa el pixel central con pendiente entre 0 y 1. Sean S_x y S_y las sumas de los 5 valores no nulos (7 en el caso que $\alpha > 0$) de la columna central de las imágenes de parciales en x y en y respectivamente. El vector normal al contorno, N, cuyo módulo indica la diferencia de intensidad existente a ambos lados del borde, viene dado por la expresión*

$$N = \frac{S_y}{\sqrt{S_x^2 + S_y^2}} [S_x, S_y] \tag{2.4}$$

2.3 Elección del pixel borde correcto (primer octante)

Hasta ahora supusimos que el contorno pasaba siempre por el pixel central. Pero ¿qué ocurriría si este dato fuera desconocido? ¿Cómo saber entonces qué pixel de la columna es el más indicado para ser etiquetado como perteneciente al borde? La respuesta será aquél cuya parcial en y sea máxima en su columna. La razón está en el perfil que toma la imagen de parciales, como se estudió al principio del presente capítulo. Es fácil demostrar que, para el caso $\alpha = 0$, si el contorno pasa por el pixel central, los 5 valores no nulos de la columna central de la imagen de parciales en y cumplen la relación

$$0 \leq f_y(0,-2) < f_y(0,-1) < f_y(0,0) > f_y(0,1) > f_y(0,2) \geq 0 \tag{2.5}$$

Para demostrarlo, veamos que, atendiendo exclusivamente a la columna central, el contorno puede cruzarla de 3 formas diferentes, como se ve en la figura 2.14 de izquierda a derecha, de izquierda a arriba, y de abajo a derecha. Vamos a estudiar cada caso por separado.

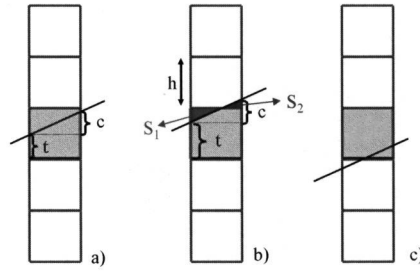


Figura 2.14: Un contorno cuya pendiente esté entre 0 y 1 puede cruzar el pixel central de tres formas distintas

Caso a: el contorno cruza sólo el pixel central En este primer caso, la columna central de la imagen original y la de la imagen de parciales en y será

$$F(0,j) = \begin{bmatrix} B \\ B \\ C \\ A \\ A \end{bmatrix}; \qquad f_y(0,j) = \frac{1}{2h} \begin{bmatrix} 0 \\ C-B \\ A-B \\ A-C \\ 0 \end{bmatrix}$$

donde $B < C < A$. Aquí es fácil ver que todas las relaciones de la expresión 2.5 se cumplen.

Caso b: el contorno cruza los pixels central y superior Cojamos ahora el caso en que el contorno cruza el pixel central de izquierda a arriba (fig. 2.14b). Las imágenes serán ahora

$$F(0,j) = \begin{bmatrix} B \\ D \\ C \\ A \\ A \end{bmatrix}; \qquad f_y(0,j) = \frac{1}{2h} \begin{bmatrix} D-B \\ C-B \\ A-D \\ A-C \\ 0 \end{bmatrix}$$

donde

$$\begin{aligned} C &= \frac{h^2 - S_1}{h^2} A + \frac{S_1}{h^2} B \\ D &= \frac{S_2}{h^2} A + \frac{h^2 - S_2}{h^2} B \end{aligned}$$

siendo S_1 el área interior al pixel $(0,0)$ sobre el contorno (triángulo verde) y S_2 el área interior al pixel $(0,-1)$ bajo el contorno (triángulo rojo). Veamos cuál de las dos parciales es más grande

$$\begin{aligned} f_y(0,-1) &= \frac{1}{2h}(C-B) = \frac{h^2 - S_1}{2h^3}(A-B) \\ f_y(0,0) &= \frac{1}{2h}(A-D) = \frac{h^2 - S_2}{2h^3}(A-B) \end{aligned}$$

Para que sea mayor la parcial del pixel $(0,0)$ debe cumplirse que $S_1 > S_2$. Teniendo en cuenta que

$$\begin{aligned} S_1 &= \frac{h}{2c}(h-t)^2 \\ S_2 &= \frac{h}{2c}(c-(h-t))^2 \end{aligned}$$

se demuestra que la condición se cumplirá cuando

$$t + \frac{c}{2} < h$$

Obsérvese en la figura que $t + c/2$ es justamente la diferencia de altura entre el contorno y el punto central del borde inferior del pixel central. Decir entonces que esta cantidad es menor que h es equivalente a decir que el trozo de contorno interior al pixel $(0,0)$ tiene mayor longitud que el trozo interior al pixel $(0,-1)$. Ya esta condición se vio en el capítulo previo, en la sección 1.3.4. Parece claro que, puestos a elegir dentro de la columna

cuál es el pixel mejor considerado para ser etiquetado como borde, debería ser aquél que cubre un trozo más largo de contorno, y acabamos de ver que es aquél que tiene la derivada parcial más alta.

Para demostrar el resto de las inecuaciones de la expresión 2.5, hay que probar que $C > D$. Para esto, baste ver que tanto C como D son de la forma

$$kA + (1-k)\,B$$

donde $k \in (0,1)$. Por lo tanto, sólo hay que comprobar que el peso de A en la expresión de C es mayor que en la de D, es decir

$$h^2 - S_1 > S_2$$

lo cual puede demostrarse que se cumplirá siempre para pendiente entre 0 y 1.

Caso c: el contorno cruza los pixels central e inferior Nos faltaría probar el tercer caso (fig. 2.14c), cuando el contorno cruza el pixel central de abajo a derecha, pero el razonamiento es similar. La parcial del pixel $(0,0)$ será máxima cuando la longitud del tramo de contorno interior a él sea mayor que la interior al pixel $(0,1)$, y esto ocurrirá cuando

$$t + \frac{c}{2} > 0$$

Finalmente, presentamos el siguiente lema:

Lema 2.4 *Sea $F_{i,j}(A,B,a,b,e)$ una imagen ideal con un contorno de primer orden cuya pendiente vale entre 0 y 1. Se cumple que, para cada columna, el pixel (i,j) por el que pasa un trozo de contorno más largo es el que cumple que*

$$0 \leq f_y(i,j-2) < f_y(i,j-1) < f_y(i,j) > f_y(i,j+1) > f_y(i,j+2) \geq 0$$

Usando este lema, en nuestra imagen F ideal con la que estamos trabajando, obtendríamos una imagen de bordes como en la figura 2.15, con un único pixel borde por columna[4]. Este dibujo se asemeja a la forma que tienen los algoritmos de dibujo de líneas sintéticos, como el de Bresenham [HEA97], para generar una línea de pixels entre dos coordenadas enteras en una imagen discreta. Estos algoritmos, para el caso de pendientes entre 0 y 1, van iterando por columnas, y eligiendo en cada una el pixel más indicado según un criterio muy parecido al que acabamos de usar. Así, obtenemos una línea de grosor 1 pixel. Esta propiedad es muy interesante porque existen algunos detectores de borde [SON98] que producen líneas de borde de grosor mayor en algunas zonas, con lo cual son menos precisos a la hora de estimar la localización exacta del contorno.

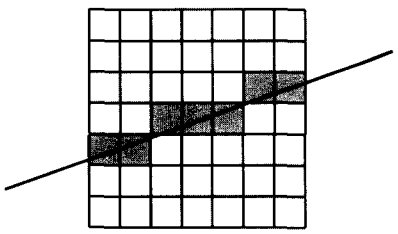


Figura 2.15: Pixels etiquetados como borde

[4]Para pendientes mayores que 1 sería un único pixel borde por fila.

2.4 Cálculo de la posición exacta del contorno (primer octante)

Ya hemos visto cómo identificar cuáles son los pixels por los que pasa el contorno, y obtener el vector normal en cada uno de ellos. Pero, ¿sería posible localizar la posición exacta por donde pasa la recta del contorno en el interior del pixel? Es decir, ¿podría deducirse el valor del parámetro t que venimos utilizando hasta ahora, a partir de los valores de la imagen original? A continuación desarrollaremos un esquema que nos permitirá obtener dicha posición.

Retornando al estudio del capítulo previo, vemos que en el caso del contorno horizontal (sección 1.3.2), la columna central en la imagen original y en la de parciales en y era

$$f_y(0,j) = \frac{1}{2h}\begin{bmatrix} 0 \\ C-B \\ A-B \\ A-C \\ 0 \end{bmatrix} = \frac{A-B}{2h^2}\begin{bmatrix} 0 \\ t \\ h \\ h-t \\ 0 \end{bmatrix}$$

donde t indicaba la posición del contorno. Vemos también que efectivamente, tal y como se indicaba en la sección 2.1.1, al sumar los tres valores no nulos de la columna desaparecía el término t y obteníamos la expresión

$$f_y(0,-1) + f_y(0,0) + f_y(0,1) = \frac{A-B}{h} = s_y$$

Sin embargo, si restamos el último valor en lugar de sumarlo, el término t quedaría aislado en el numerador, obteniendo la siguiente expresión:

$$f_y(0,-1) + f_y(0,0) - f_y(0,1) = \left(\frac{A-B}{h}\right)\frac{t}{h} = s_y\frac{t}{h}$$

con lo cual tendremos, después de dividir por s_y, la altura relativa, t/h, a la que se encuentra el contorno. Vamos entonces a estudiar el caso general, y veremos que mediante sumas y restas de los valores de f_y podemos deducir la posición en todos los casos.

2.4.1 Caso general con $\alpha = 0$

Sea de nuevo $F_{i,j}(A,B,a,b,e)$ una imagen ideal con un contorno de pendiente entre 0 y 1, que atraviesa el pixel central (ecuación 2.2). Atendiendo a la forma en la que el contorno cruza dicho pixel, volvemos a tener 3 casos diferentes (fig. 2.14).

Caso a: el contorno cruza sólo el pixel central Empecemos por el primer caso (a), en donde el contorno cruza el pixel central de izquierda a derecha. La imagen original y la de parciales en y será

$$F(0,j) = \begin{bmatrix} B \\ B \\ C \\ A \\ A \end{bmatrix}; \qquad f_y(0,j) = \frac{1}{2h}\begin{bmatrix} 0 \\ C-B \\ A-B \\ A-C \\ 0 \end{bmatrix}$$

donde

$$C = \frac{th + \frac{1}{2}ch}{h^2}A + \frac{h^2 - th - \frac{1}{2}ch}{h^2}B = B + (A-B)\frac{t + \frac{1}{2}c}{h} \tag{2.6}$$

Si realizamos la operación que hicimos antes con los tres valores no nulos de f_y obtenemos

$$f_y(0,-1) + f_y(0,0) - f_y(0,1) = \frac{1}{h}(C - B) = \left(\frac{A-B}{h}\right)\frac{t+\frac{1}{2}c}{h} = s_y\frac{d}{h}$$

donde d es la diferencia de altura entre el contorno y el punto central del borde inferior del pixel $(0,0)$, como se ve en la figura 2.16a.

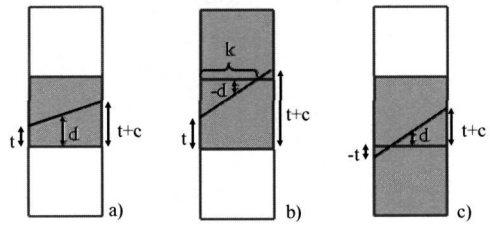


Figura 2.16: Posición del contorno dentro del pixel.

Caso b: el contorno cruza los pixels central y superior Tomemos ahora el caso de la figura 2.14b. La imagen y su parcial son:

$$F(0,j) = \begin{bmatrix} B \\ D \\ C \\ A \\ A \end{bmatrix}; \qquad f_y(0,j) = \frac{1}{2h}\begin{bmatrix} D-B \\ C-B \\ A-D \\ A-C \\ 0 \end{bmatrix}$$

donde

$$\begin{aligned} C &= \frac{h^2 - \frac{1}{2}k(h-t)}{h^2}A + \frac{\frac{1}{2}k(h-t)}{h^2}B \\ D &= \frac{\frac{1}{2}(h-k)(t+c-h)}{h^2}A + \frac{h^2 - \frac{1}{2}(h-k)(t+c-h)}{h^2}B \end{aligned}$$

siendo k la distancia horizontal que aparece en la figura 2.16b, y cuya expresión es

$$k = \frac{h(h-t)}{c}$$

Sustituyendo este valor de k, nos queda que

$$\begin{aligned} C &= A - \frac{(h-t)^2}{2ch}(A-B) \\ D &= 2B - A + \left(\frac{(h-t)^2}{2ch} + \frac{c+2t}{2h}\right)(A-B) \end{aligned} \tag{2.7}$$

Como ahora el contorno atraviesa dos pixels de la misma columna, aparecen hasta 4 valores no nulos en la columna central de la imagen de parciales. Calculemos entonces la siguiente expresión: sumar las parciales

de los dos pixels sobre el borde, y restar los dos de debajo. A esta expresión la llamaremos r_y, y su valor es exactamente el mismo que antes, como podemos ver:

$$\begin{aligned} r_y &= f_y(0,-2) + f_y(0,-1) - f_y(0,0) - f_y(0,1) = \frac{1}{h}(C + D - A - B) = \\ &= \frac{1}{h}\left(B - A + \frac{c+2t}{2h}(A-B)\right) = \left(\frac{A-B}{h}\right)\frac{t+\frac{c}{2}-h}{h} = \\ &= s_y\frac{d}{h} \end{aligned} \tag{2.8}$$

donde ahora d es la diferencia de altura entre el contorno y el punto central del borde superior del pixel[5]. ¿Por qué antes se medía d desde la base del pixel central, y ahora es desde el borde superior? La diferencia está en qué pixel comienza la resta en la expresión para r_y, ya que d siempre va a medirse desde la frontera de los dos pixels donde se produce el cambio de signo. En efecto, en el primer caso donde el contorno sólo cruza el pixel $(0,0)$, la parcial de este pixel aparecía con signo + y luego restábamos la del $(0,1)$, y por ello d se media desde el borde entre estos dos pixels. Pero ahora, la parcial del pixel $(0,-1)$ tiene signo + y la del $(0,0)$ aparece con signo -, por lo que d se mide ahora desde la frontera entre estos dos pixels.

Caso c: el contorno cruza los pixels central e inferior Nos faltaría el caso de la figura 2.14c, pero vemos que es equivalente al caso (b), siempre y cuando desplacemos la expresión de r_y un pixel hacia abajo, es decir

$$r_y = f_y(0,-1) + f_y(0,0) - f_y(0,1) - f_y(0,2) = s_y\frac{d}{h} \tag{2.9}$$

sólo que ahora d se mide también en vertical pero desde la frontera entre los pixels $(0,0)$ y $(0,1)$ (figura 2.16c.

Distinción entre los casos b y c Sin embargo, el problema que se plantea ahora es el siguiente: ¿cómo podemos distinguir en cuál de los tres casos estamos, para saber qué expresión de r_y aplicar? Veamos primero cómo diferenciar entre los casos b y c. Sabemos que el contorno atraviesa el pixel central, y que la parcial de este pixel, $f_y(0,0)$, es mayor que la de sus dos vecinos en esa columna. Nos faltaría saber cuál de esos dos vecinos es el que también es atravesado por el contorno, para saber qué expresión aplicar para r_y (ecuación 2.8 ó 2.9), y para ello demostraremos que será el que tenga mayor parcial en y que el otro. Para ello observemos lo siguiente: tomemos el caso (b), donde $t + c > h$. Queremos demostrar que efectivamente en este caso se cumple que

$$f_y(0,-1) > f_y(0,1) \Rightarrow C > \frac{A+B}{2}$$

Atendiendo a la expresión de C obtenida antes (ecuación 2.7), y sabiendo que $t + c > h$, deducimos que

$$C > A - \frac{c}{2h}(A-B)$$

y como $c < h$, entonces

$$C > A - \frac{1}{2}(A-B) = \frac{A+B}{2}$$

con lo cual queda demostrado que, si el contorno atraviesa el pixel $(0,-1)$, entonces se cumple que $f_y(0,-1) > f_y(0,1)$. Otra forma de verlo es que, para que se cumpla que $t + c > h$ ha de cumplirse que $t + c/2 > h/2$, puesto que $c < h$. Esto significa que el contorno está más cercano al pixel $(0,-1)$ que al pixel $(0,1)$, y que por

[5] En realidad es una distancia con signo, ya que si $t + c/2 < h$, entonces $d < 0$.

tanto, el área interior al pixel $(0,0)$ bajo el contorno (correspondiente al valor A) será mayor que el área sobre el contorno (correspondiente al valor B), con lo cual $C > (A+B)/2$.

Para el caso (c), el razonamiento es similar, con lo cual si el contorno atraviesa el pixel $(0,1)$, entonces $f_y(0,1) > f_y(0,-1)$. Esto significa que, atendiendo a qué parcial sea mayor, usaremos una expresión de r_y u otra para obtener la posición exacta del contorno.

Distinción con el caso a ¿Y qué ocurre con el caso (a)? Pues vamos a ver que, atendiendo también a cuál de los dos vecinos verticales del pixel central tenga mayor parcial, podemos usar la misma regla para conocer la expresión de r_y que nos dará el resultado correcto. Efectivamente, supongamos que $f_y(0,-1) > f_y(0,1)$. Entonces, atendiendo a la expresión de C obtenida para este caso (ecuación 2.6), se cumple que

$$C > \frac{A+B}{2} \Rightarrow t + \frac{c}{2} > \frac{h}{2}$$

Es decir, el contorno está más cerca del pixel $(0,-1)$ que del $(0,1)$. En ese caso, si aplicamos la expresión de r_y correspondiente obtenemos

$$r_y = f_y(0,-2) + f_y(0,-1) - f_y(0,0) - f_y(0,1) = \frac{1}{h}(C-A) = \left(\frac{A-B}{h}\right)\frac{t+\frac{c}{2}-h}{h} = s_y\frac{d}{h}$$

con d medido desde la frontera entre los pixels $(0,-1)$ y $(0,0)$.

Suponiendo ahora lo contrario, $f_y(0,1) > f_y(0,-1)$, tenemos que

$$C < \frac{A+B}{2} \Rightarrow t + \frac{c}{2} < \frac{h}{2}$$

con lo cual el contorno está más cerca del pixel $(0,1)$ que del $(0,-1)$. Aplicando la expresión de r_y alternativa, obtenemos idéntico resultado

$$r_y = f_y(0,-1) + f_y(0,0) - f_y(0,1) - f_y(0,2) = \frac{1}{h}(C-B) = \left(\frac{A-B}{h}\right)\frac{t+\frac{1}{2}c}{h} = s_y\frac{d}{h}$$

con d esta vez medido desde la frontera entre los pixels $(0,0)$ y $(0,1)$.

Con todo esto se concluye que no es preciso averiguar primero en cuál de los tres casos estamos. Simplemente atendiendo a cuál de las parciales de los dos vecinos es mayor, aplicaremos la expresión de r_y correspondiente.

Medición de la distancia desde el centro del pixel Para que la distancia d no esté medida unas veces con respecto a la base del pixel central, y otras con respecto a su frontera superior, llamemos p a una nueva cantidad que represente la diferencia de altura entre el contorno y el centro geométrico del pixel, como se ve en la figura 2.17. Así, la expresión para p será

$$p = \begin{cases} -h/2 - d & \text{si } f_y(0,-1) > f_y(0,1) \\ h/2 - d & \text{si } f_y(0,-1) < f_y(0,1) \end{cases}$$

siendo $0 < p < h/2$ cuando el contorno pasa por debajo del centro del pixel, y $-h/2 < p < 0$ cuando pasa por encima.

Con todo lo anterior ya podemos enunciar el próximo lema:

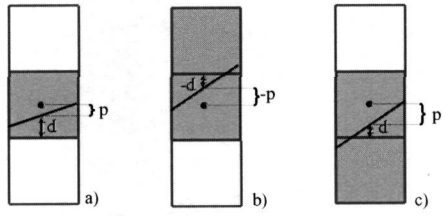


Figura 2.17:

Lema 2.5 *Sea $F_{i,j}(A, B, a, b, e)$ una imagen ideal con un contorno de primer orden que atraviesa el pixel central con pendiente entre 0 y 1, y que cumple que $f_y(0,-1) < f_y(0,0) > f_y(0,1)$, para $\alpha = 0$. Sea r_y la expresión siguiente*

$$r_y = \sum_{j=m-2}^{j=m-1} f_y(0,j) - \sum_{j=m}^{j=m+1} f_y(0,j)$$

donde

$$m = \begin{cases} 0 & \text{si } f_y(0,-1) \geq f_y(0,1) \\ 1 & \text{si } f_y(0,-1) < f_y(0,1) \end{cases}$$

El desplazamiento exacto del contorno dentro del pixel viene dado por la ecuación

$$\frac{p}{h} = \frac{2m-1}{2} - \frac{r_y}{s_y} \tag{2.10}$$

donde s_y es la suma de los 5 valores no nulos de la columna central de la imagen de parciales en y, h es la longitud del pixel, y p es la diferencia de altura entre el contorno y el centro geométrico del pixel $(0,0)$.

2.4.2 Caso general con $\alpha > 0$

Cuando $\alpha > 0$, la situación es un poco más complicada, debido a que entran muchos más pixels en juego. Hay que tener en cuenta que la expresión de r_y usa solamente los valores de las parciales de la columna central, $f_y(0,j)$. Con $\alpha = 0$, para obtener estas parciales sólo se tienen en cuenta los pixels de la columna central, $F(0,j)$. Sin embargo, cuando $\alpha > 0$, hay que usar también las columnas -1 y 1 de la imagen. Por ejemplo, el mismo caso simple de la figura 2.16a se subdividiría en 4 subcasos diferentes, que son los casos 3, 4, 7 y 8 de la figura 2.9. Si tomamos de estos últimos el caso 3, y evaluásemos la expresión r_y obtendríamos

$$r_y = \frac{t + \frac{c}{2}}{h^2}(A - B) - \alpha(t + 2c - h)^2 \frac{3}{8ch^2}(A - B)$$

Puede verse que aparece un término escalado por α, a partir del cual se haría bastante complicado obtener el valor de t. Analicemos en una situación más general el porqué de esta nueva expresión.

Cojamos por ejemplo el caso donde $F_y(0,-1) > F_y(0,1)$, lo cual ya hemos visto que implica que $t+c/2 > h/2$. La situación alrededor del pixel central puede verse en la figura 2.18. Estudiemos ahora la expresión de cada una de las parciales en y de la columna central. Recordando la ecuación para las parciales en y

$$\begin{aligned} F_y(i,j) &= (1-\alpha)\left(\frac{F_{i,j+1} - F_{i,j-1}}{2h}\right) + \frac{\alpha}{2}\left(\frac{F_{i+1,j+1} - F_{i+1,j-1} + F_{i-1,j+1} - F_{i-1,j-1}}{2h}\right) = \\ &= \frac{F_{i,j+1} - F_{i,j-1}}{2h} + \frac{\alpha}{4h}(F_{i+1,j+1} - F_{i+1,j-1} + F_{i-1,j+1} - F_{i-1,j-1} - 2F_{i,j+1} + 2F_{i,j-1}) \end{aligned}$$

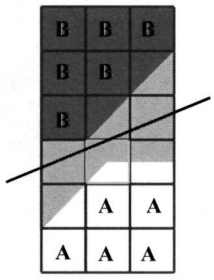


Figura 2.18: Situación con $t + c/2 > h/2$

con lo cual nos salen en total 6 valores no nulos en esta columna, en lugar de 4 como ocurría cuando $\alpha = 0$ (puede verse cómo los valores de los pixels extremos de la columna, $F_y(0,-3)$ y $F_y(0,2)$ son proporcionales al valor de α). Esos valores son los siguientes:

$$\begin{aligned}
F_y(0,-3) &= \frac{\alpha}{4h}(F_{1,-2} - B) \\
F_y(0,-2) &= \frac{F_{0,-1} - B}{2h} + \frac{\alpha}{4h}(F_{1,-1} - F_{1,-3} + F_{-1,-1} - F_{-1,-3} - 2F_{0,-1} + 2B) \\
F_y(0,-1) &= \frac{F_{0,0} - B}{2h} + \frac{\alpha}{4h}(F_{1,0} - F_{1,-2} + F_{-1,0} - F_{-1,-2} - 2F_{0,0} + 2B) \\
F_y(0,0) &= \frac{A - F_{0,-1}}{2h} + \frac{\alpha}{4h}(F_{1,1} - F_{1,-1} + F_{-1,1} - F_{-1,-1} + 2F_{0,-1} - 2A) \\
F_y(0,1) &= \frac{A - F_{0,0}}{2h} + \frac{\alpha}{4h}(F_{1,2} - F_{1,0} + F_{-1,2} - F_{-1,0} + 2F_{0,0} - 2A) \\
F_y(0,2) &= \frac{\alpha}{4h}(A - F_{-1,1})
\end{aligned}$$

Si evaluamos la expresión para r_y asociada a este caso resulta

$$r_y = F_y(0,-2) + F_y(0,-1) - F_y(0,0) - F_y(0,1) = T_1 + \frac{\alpha}{4h}T_2$$

donde

$$T_1 = \frac{F_{0,-1} + F_{0,0} - A - B}{h}$$

es justamente lo que obtendríamos si estuviéramos en el caso $\alpha = 0$, y

$$\begin{aligned}
T_2 = {} & 4(A+B) - 4(F_{0,0} + F_{0,-1}) + 2(F_{-1,0} + F_{1,0} + F_{-1,-1} + F_{1,-1}) - \\
& - (F_{-1,1} + F_{1,1} + F_{-1,-2} + F_{-1,2} + F_{1,-2} + F_{1,2} + F_{-1,-3} + F_{1,-3})
\end{aligned}$$

es un término que depende de los pixels de la vecindad. Teniendo en cuenta que, atendiendo a la figura 2.18 podemos considerar que $F_{1,-3} = F_{-1,-3} = F_{-1,-2} = B$ y que $F_{-1,2} = F_{1,2} = F_{1,1} = A$, nos queda que

$$T_2 = A + B - 4(F_{0,0} + F_{0,-1}) + 2(F_{-1,0} + F_{1,0} + F_{-1,-1} + F_{1,-1}) - (F_{-1,1} + F_{1,-2})$$

Por otro lado, aún podríamos simplificar más la expresión de T_2 si considerásemos también dentro de r_y los valores de $F_y(0,-3)$ y $F_y(0,2)$. En ese caso, llamemos a la nueva expresión R_y, cuyo valor será

$$R_y = F_y(0,-3) + F_y(0,-2) + F_y(0,-1) - F_y(0,0) - F_y(0,1) - F_y(0,2) = T_1 + \frac{\alpha}{4h}T_2$$

donde T_1 es igual que antes, y

$$T_2 = 2\left(F_{-1,0} + F_{1,0} + F_{-1,-1} + F_{1,-1}\right) - 4\left(F_{0,0} + F_{0,-1}\right)$$

A partir de esta expresión para R_y sería muy difícil tratar de obtener el valor de t. Pero lo que sí podemos hacer es evaluar por separado dicho término T_2 a partir de los pixels de la imagen original, escalarlo por $\alpha/4h$, y restárselo al valor de R_y para obtener T_1, a partir del cual podemos obtener d/h como se hacía en la sección anterior. Con esto llegamos al próximo lema

Lema 2.6 *Sea $F_{i,j}(A, B, a, b, e)$ una imagen ideal con un contorno de primer orden que atraviesa el pixel central con pendiente entre 0 y 1, y que cumple que $F_y(0,-1) < F_y(0,0) > F_y(0,1)$. Sea R_y la expresión siguiente*

$$R_y = \sum_{j=m-3}^{j=m-1} F_y(0,j) - \sum_{j=m}^{j=m+2} F_y(0,j)$$

donde

$$m = \begin{cases} 0 & \text{si } F_y(0,-1) \geq F_y(0,1) \\ 1 & \text{si } F_y(0,-1) < F_y(0,1) \end{cases}$$

El desplazamiento exacto del contorno dentro del pixel viene dado por la ecuación

$$\frac{p}{h} = \frac{2m-1}{2} - \frac{1}{S_y}\left(R_y - \frac{\alpha}{2h}T\right) \tag{2.11}$$

siendo

$$T = F_{-1,m} + F_{1,m} + F_{-1,m-1} + F_{1,m-1} - 2\left(F_{0,m} + F_{0,m-1}\right)$$

y donde S_y es la suma de los 7 valores no nulos de la columna central de la imagen de parciales en y, h es la longitud del pixel, y p es la diferencia de altura entre el contorno y el centro geométrico del pixel $(0,0)$.

Es evidente que, aunque el razonamiento se haya hecho sólo para el caso cuando $t + c/2 > h$, el resultado para el otro caso es idéntico si desplazamos todas las expresiones 1 pixel hacia abajo.

2.5 Generalización para el resto de octantes

Todo el estudio anterior se ha realizado para contornos de pendiente entre 0 y 1. Ahora extenderemos por simetría la situación para localizar contornos con cualquier pendiente. Recordemos que para el primer octante, el vector normal al contorno viene dado por la expresión 2.4 y su posición relativa dentro del pixel viene dado por la expresiones 2.10 ó 2.11 según si α es nulo o no respectivamente. Este caso se corresponde con el número 1 de la figura 2.19.

Estudiemos ahora el caso 8, cuando la pendiente es negativa, entre 0 y -1. Vemos que la diferencia está en que las parciales en x serán ahora negativas, con lo cual $S_x < 0$. Todo lo demás es exactamente igual. El único cambio que habría que hacer es usar el valor absoluto para calcular el coeficiente escalar del vector normal, es decir

$$N = \frac{|S_y|}{\sqrt{S_x^2 + S_y^2}}\left[S_x, S_y\right]$$

Analicemos ahora cuando las intensidades a ambos lados del contorno son al revés, es decir, A por encima y B por debajo, siendo $A > B$ (casos 4 y 5). Las parciales en y ahora serán negativas, con lo cual $S_y < 0$. La

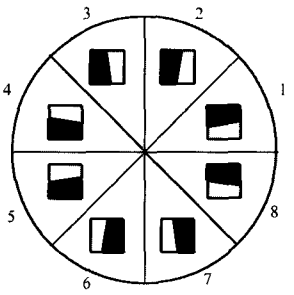


Figura 2.19: El contorno de una imagen pertenecerá siempre a uno de estos 8 octantes.

fórmula para el desplazamiento, R_y, funciona exactamente igual, sólo que ahora la condición para elegir una expresión u otra ha de realizarse según los valores absolutos de las parciales, es decir

$$\begin{aligned} R_y &= \sum_{j=m-3}^{j=m-1} F_y(0,j) - \sum_{j=m}^{j=m+2} F_y(0,j) \\ m &= \begin{cases} 0 & \text{si } |F_y(0,-1)| \geq |F_y(0,1)| \\ 1 & \text{si } |F_y(0,-1)| < |F_y(0,1)| \end{cases} \end{aligned}$$

Con esto, el esquema final para estos 4 octantes viene dado por el siguiente lema:

Lema 2.7 *Sea $F_{i,j}(A,B,a,b,e)$ una imagen ideal con un contorno de primer orden que atraviesa el pixel central, y que cumple que*

$$\begin{aligned} |F_y(0,0)| &> |F_x(0,0)| \\ |F_y(0,1)| &< |F_y(0,0)| > |F_y(0,-1)| \end{aligned}$$

El vector normal al contorno, N, cuyo módulo indica la diferencia de intensidad existente a ambos lados del borde, viene dado por la expresión

$$N = \frac{|S_y|}{\sqrt{S_x^2 + S_y^2}} [S_x, S_y]$$

y su posición vertical relativa dentro del pixel viene dada por la expresión

$$\frac{p}{h} = \frac{2m-1}{2} - \frac{1}{S_y}\left(R_y - \frac{\alpha}{2h}T\right)$$

donde

$$\begin{aligned} S_x &= \sum_{j=-n}^{n} F_x(0,j) \\ S_y &= \sum_{j=-n}^{n} F_y(0,j) \\ n &= \begin{cases} 2 & \text{si } \alpha = 0 \\ 3 & \text{si } \alpha > 0 \end{cases} \end{aligned}$$

$$
\begin{aligned}
R_y &= \sum_{j=m-3}^{j=m-1} F_y(0,j) - \sum_{j=m}^{j=m+2} F_y(0,j) \\
T &= F_{-1,m} + F_{1,m} + F_{-1,m-1} + F_{1,m-1} - 2\left(F_{0,m} + F_{0,m-1}\right) \\
m &= \begin{cases} 0 & \text{si } |F_y(0,-1)| \geq |F_y(0,1)| \\ 1 & \text{si } |F_y(0,-1)| < |F_y(0,1)| \end{cases}
\end{aligned}
$$

Finalmente, para los otros 4 octantes, se cumple que $|F_x(0,0)| > |F_y(0,0)|$, con lo cual habría que volver a repetir todo el razonamiento de este capítulo, intercambiando los dos ejes de coordenadas, y trabajando con la fila central en lugar de con la columna. Hay que tener en cuenta que ahora la posición exacta del contorno vendrá dada en horizontal, desde el centro geométrico del pixel $(0,0)$. El siguiente lema indica el esquema para estos casos:

Lema 2.8 *Sea $F_{i,j}(A,B,a,b,e)$ una imagen ideal con un contorno de primer orden que atraviesa el pixel central, y que cumple que*

$$
\begin{aligned}
|F_x(0,0)| &> |F_y(0,0)| \\
|F_x(0,1)| &< |F_x(0,0)| > |F_x(0,-1)|
\end{aligned}
$$

El vector normal al contorno, N, cuyo módulo indica la diferencia de intensidad existente a ambos lados del borde, viene dado por la expresión

$$
N = \frac{|S_x|}{\sqrt{S_x^2 + S_y^2}} [S_x, S_y]
$$

y su posición horizontal relativa dentro del pixel viene dada por la expresión

$$
\frac{p}{h} = \frac{2m-1}{2} - \frac{1}{S_x}\left(R_x - \frac{\alpha}{2h}T\right)
$$

donde

$$
\begin{aligned}
S_x &= \sum_{i=-n}^{n} F_x(i,0); \\
S_y &= \sum_{i=-n}^{n} F_y(i,0) \\
n &= \begin{cases} 2 & \text{si } \alpha = 0 \\ 3 & \text{si } \alpha > 0 \end{cases} \\
R_x &= \sum_{i=m-3}^{i=m-1} F_x(i,0) - \sum_{i=m}^{i=m+2} F_x(i,0) \\
T &= F_{m,-1} + F_{m,1} + F_{m-1,-1} + F_{m-1,1} - 2\left(F_{m,0} + F_{m-1,0}\right) \\
m &= \begin{cases} 0 & \text{si } |F_x(-1,0)| \geq |F_x(1,0)| \\ 1 & \text{si } |F_x(-1,0)| < |F_x(1,0)| \end{cases}
\end{aligned}
$$

Para saber en cuál de los 8 octantes nos encontramos, bastará ver los signos de ambas parciales, y ver cuál de ellas es mayor en valor absoluto.

2.6 Algoritmo para la localización de contornos

A partir de los lemas expuestos anteriormente, ya estamos en disposición de desarrollar un algoritmo para tratar de detectar la posición y orientación de los contornos en una imagen. En primer lugar, habrá que detectar qué pixels son etiquetados como borde. Para ello, obtendremos las imágenes de parciales F_x y F_y a partir de las máscaras H_x y H_y, eligiendo el valor de α que se quiera aplicar, y nos centraremos en aquellos pixels (i, j) donde

$$F_x^2(i,j) + F_y^2(i,j) > \delta^2$$

donde δ es un umbral que habrá que prefijar. Aunque ya expusimos en el capítulo anterior que esta expresión del gradiente no era exacta, sí que puede usarse como aproximación para saber si existe un salto de intensidad grande en las cercanías del pixel.

Pero para catalogar un pixel como borde no bastará que cumpla esta condición. Además estamos interesados en localizar perfiles como el que vimos en la sección 2.1.1, es decir, zonas donde el valor absoluto de la derivada parcial crezca, llegue a un máximo, y vuelva a decrecer. Por lo tanto, la segunda condición que ha de cumplir un pixel para ser borde es que

$$|F_y(i,j-2)| < |F_y(i,j-1)| < |F_y(i,j)| > |F_y(i,j+1)| > |F_y(i,j+2)|$$

en el caso en que $|F_y(i,j)| > |F_x(i,j)|$, o bien que

$$|F_x(i-2,j)| < |F_x(i-1,j)| < |F_x(i,j)| > |F_x(i+1,j)| > |F_x(i+2,j)|$$

en el caso en que $|F_y(i,j)| < |F_x(i,j)|$

Una vez tenemos los pixels identificados, aplicaremos alrededor de cada uno el esquema desarrollado en las secciones anteriores para obtener los parámetros del contorno (posición y orientación). Esto significa que, para cada pixel borde, el método usará los valores de un entorno centrado en dicho pixel, cuya forma variará en función de si α es nulo o no, y de qué parcial es mayor. En las figuras 2.20 y 2.21 se muestran los entornos usados para el caso en el que $|f_y| > |f_x|$ cuando α es nulo o no respectivamente. Puede apreciarse cómo el caso $\alpha > 0$ es en el que más pixels vecinos se están usando, siendo el entorno una ventana rectangular de 3 pixels de ancho y 9 de alto.

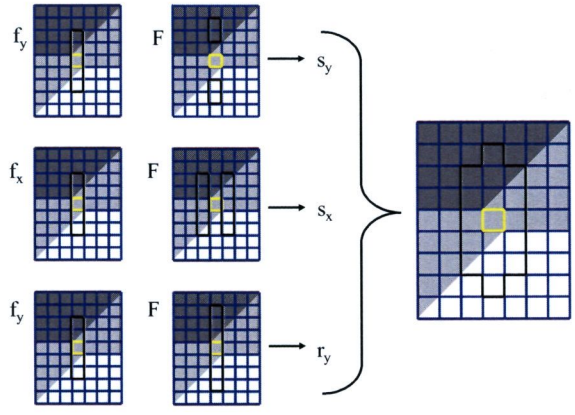


Figura 2.20: Pixels utilizados en la imagen original para el cálculo de la posición y orientación del contorno en el pixel central cuando $\alpha = 0$

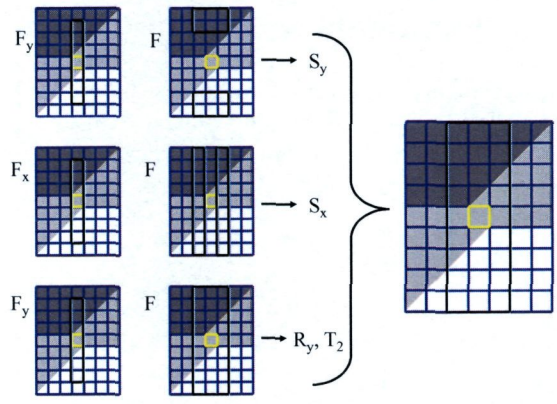


Figura 2.21: Pixels utilizados en la imagen original para el cálculo de la posición y orientación del contorno en el pixel central cuando $\alpha > 0$

Hay tres condiciones que deben cumplirse dentro de dicho entorno, o al menos acercarse lo más posible, para que la situación alrededor de esos pixels sea lo más parecido a la situación ideal sobre la que se ha desarrollado el método:

1) Dentro de ese entorno, el contorno puede ser aproximado como una línea recta.

2) Dentro de ese entorno, no debe aparecer ningún otro contorno diferente.

3) Dentro de ese entorno, la intensidad a ambos lados del contorno debe ser lo más homogénea posible.

Lógicamente, es difícil que las tres condiciones se cumplan exhaustivamente en todos los pixels borde, pero también es bastante probable que se cumplan en un grado alto. La primera condición fallará en zonas donde haya esquinas, o bordes con una curvatura muy alta. La segunda fallará en zonas donde haya dos bordes muy cercanos entre sí. La tercera fallará en los bordes de zonas con textura muy fina. En los demás casos, las condiciones se cumplirán. Por otro lado, aunque el entorno abarque bastantes pixels alrededor del pixel central, hay que tener en cuenta que los más alejados del centro son los que menos peso llevan en las expresiones, por lo que no es muy grave que las condiciones anteriores fallen un poco más en dichos pixels, mientras se cumplan con más exactitud en las cercanías del pixel central.

Hay otra condición que también asumimos, y es que la imagen no tenga ruido, o que tenga muy poco. En capítulos posteriores abordaremos este problema, pero por ahora trabajaremos con imágenes lo más limpias posibles.

El pseudo-código de nuestro algoritmo podría ser el siguiente:

Funcion Contornos (delta)

```
Calcular Fx, Fy
Para todos los pixels (i,j) de la imagen
   Si Fx(i,j)*Fx(i,j) + Fy(i,j)*Fy(i,j) < delta * delta Continuar
   Si |Fy(i,j)| > |Fx(i,j)| Entonces
      Si |Fy(i,j)| no es maximo en su columna Continuar
      Calcular Vector Normal y Posicion segun Lema 7
         Si No
      Si |Fx(i,j)| no es maximo en su fila Continuar
      Calcular Vector Normal y Posicion segun Lema 8
   Fin Si
```

```
Fin Para
```

2.7 Ejemplos

Veamos a continuación algunos ejemplos de aplicación de nuestro método sobre varias imágenes, en los que se podrán comparar los resultados obtenidos por nuestro método con los del método tradicional de convolución con las máscaras H_x y H_y. Para ello usaremos tanto imágenes sintéticas como reales.

2.7.1 Contorno recto ideal

Comencemos por aplicar nuestro método sobre imágenes ideales, generadas sintéticamente, con un único borde que separa dos áreas de distinta intensidad, como se ve en la figura 2.22. Se ha escogido un valor de intensidad 0 para la zona negra y 255 para la blanca. Para los pixels que son atravesados por el contorno, se ha usado nuestra hipótesis para el modelo de adquisición, calculando su intensidad en función de las áreas relativas de cada color dentro del pixel.

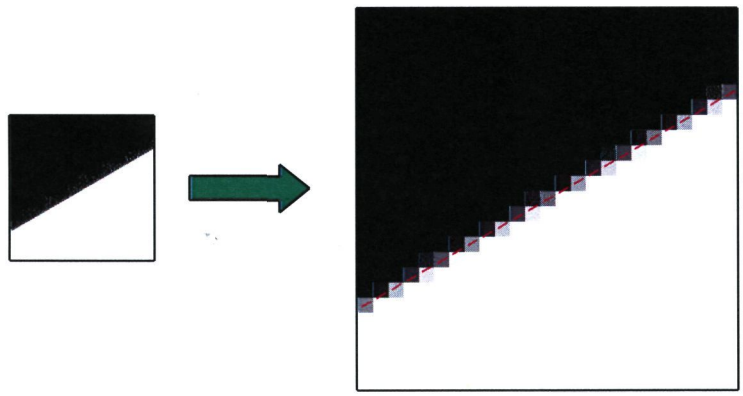


Figura 2.22: Contorno recto ideal de 30 grados generado sintéticamente. Los segmentos de color muestran la orientación y localización del contorno estimado por nuestro método para cada pixel.

Para poder visualizar el resultado, se ha generado una imagen ampliada, donde cada pixel de la imagen original ocupa un cuadrado de n x n pixels. Y en dicha imagen, se ha mostrado dentro de cada pixel borde un segmento recto de color que indica la orientación y localización estimadas para el contorno dentro de dicho pixel. Puede observarse en dicha figura cómo, visualmente, los segmentos rectos parecen estar bien alineados unos con otros. Además, como ya se comentó anteriormente, en cada columna aparece un único pixel etiquetado como borde (ya que la figura muestra el caso para el primer octante).

En total se han probado 8 imágenes diferentes (una en cada octante), asignando un valor de pendiente arbitrario en cada caso. Con nuestro esquema, y de forma independiente al valor de α que escojamos, en todos y cada uno de los pixels etiquetados como borde el módulo del vector normal que se obtiene es exactamente 255. La orientación de dicho vector es exactamente la usada para generar la imagen. Y el desplazamiento dentro del pixel es exactamente el que debe salir.

Por el contrario, al aplicar el método tradicional usando $\alpha = 0$ existen errores tanto en módulo como en dirección (y no existe información para el desplazamiento). En la tabla 2.1 se muestra, para la imagen ejemplo usada en cada octante, la pendiente del contorno que se generó, los valores medio, mínimo y máximo y la

	Error del módulo				Error de la dirección			
Grados	Media	Desv.	Mín.	Max.	Media	Desv.	Mín.	Max.
5	3.5	0.8	0.1	3.8	4.4	1.1	0.0	4.9
55	78.5	19.3	38.2	99.8	6.4	2.0	2.4	9.0
105	25.4	9.8	1.2	34.3	9.5	4.0	0.6	13.1
155	54.7	20.3	16.8	79.2	9.8	4.4	1.9	15.2
200	40.2	15.4	12.1	59.5	10.2	4.5	2.2	15.8
240	68.2	20.8	25.0	91.9	8.6	3.4	2.0	12.6
280	12.7	4.1	2.1	15.3	7.7	2.6	1.4	9.4
320	86.7	15.7	52.1	104.2	3.5	0.8	1.8	4.7

Tabla 2.1: Errores cometidos por el método tradicional, usando alfa nulo, al obtener el contorno de una recta en distintos octantes

	Error del módulo				Error de la dirección			
Grados	Media	Desv.	Mín.	Max.	Media	Desv.	Mín.	Max.
5	1.2	1.0	0.2	6.3	0.1	0.2	0.0	1.1
55	11.0	5.4	2.1	17.9	1.4	0.1	1.1	1.6
105	6.9	3.3	0.4	15.4	0.7	0.8	0.0	2.6
155	11.5	6.0	0.2	19.1	1.5	0.7	0.5	2.6
200	9.4	4.8	1.0	15.8	1.1	0.9	0.0	2.6
240	11.8	5.9	0.4	19.2	1.6	0.3	1.1	2.1
280	3.8	1.7	0.8	8.6	0.3	0.5	0.0	1.7
320	9.6	5.1	0.4	16.3	0.9	0.2	0.4	1.1

Tabla 2.2: Errores cometidos por el método tradicional, usando alfa no nulo, al obtener el contorno de una recta en distintos octantes

desviación típica, tanto para el módulo como para la dirección obtenida en cada pixel borde. Los valores de error del módulo están medidos en unidades de intensidad, y los de la dirección en grados.

Dichos errores son a veces muy grandes. Por ejemplo, para el caso del contorno de 320 grados, el error medio cometido en el módulo del gradiente es mayor que 86 unidades (34% de error relativo) llegando a encontrar un pixel con un error de más de 104 unidades (41% de error relativo). En cuanto a la dirección del vector, en el caso del contorno de 200 grados, el error medio cometido es mayor de 10 grados, llegando a encontrar un pixel cuya dirección tiene un error de más de 15 grados.

Para $\alpha = 0.5$ los errores disminuyen aunque siguen presentes, como se muestra en la tabla 2.2. El mayor error encontrado para el módulo es de 19.2 unidades, y para la orientación es de 2.6 grados.

2.7.2 Contorno circular ideal

A continuación probamos con la imagen de un círculo sintético, de radio 20 pixels, con intensidad 255 en el interior y 0 en el exterior, generada también según la hipótesis de nuestro modelo de adquisición. Puede apreciarse en la figura 2.23 como en la vecindad local de un pixel, el contorno puede ser aproximado por un borde recto sin demasiado error. Nuestro esquema produce un borde alrededor de la circunferencia, donde puede verse que el error de la pendiente es muy pequeño (figura 2.23b). Si aplicásemos el esquema tradicional, el error es bastante mayor, sobre todo en las zonas de pendientes oblicuas (figura 2.23a).

En la tabla 2.3 se muestran los errores cometidos en módulo (unidades de intensidad) y dirección (grados) con ambos métodos, probando cada uno con dos valores distintos de α (0 y 0.5). Vemos que nuestro método

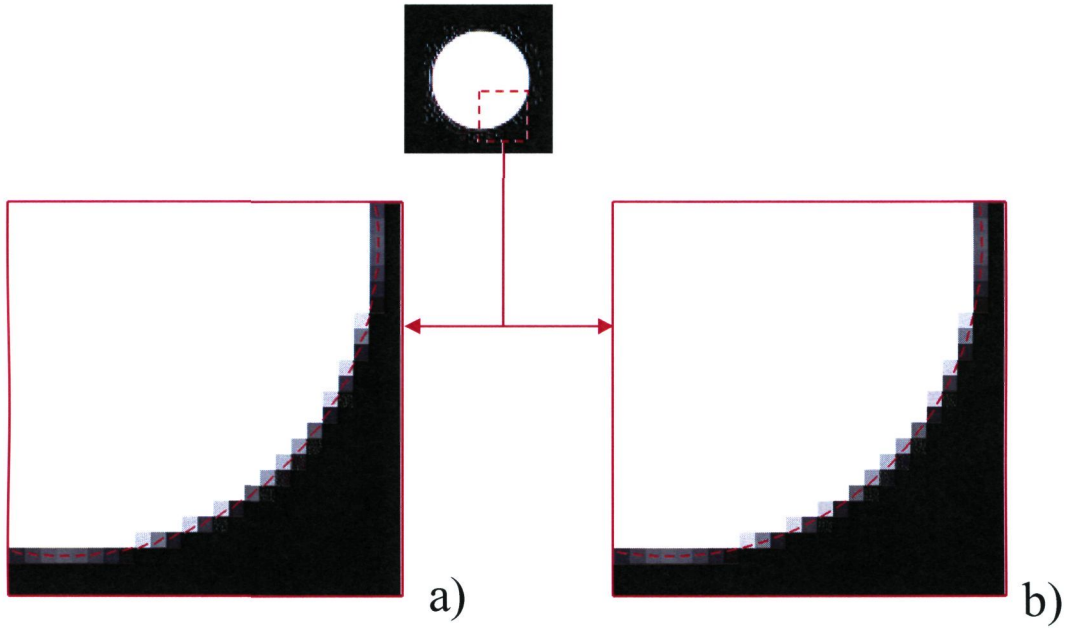


Figura 2.23: Imagen de un círculo ideal generado sintéticamente. a) Resultados del método tradicional. b) Resultados de nuestro método. Para la imagen (a) se han tomado "prestados" los valores de desplazamiento obtenidos por nuestro método.

consigue una precisión total en cuanto al módulo, y para la dirección también se obtienen valores bastante precisos, siendo el error medio 0.05 grados y el error máximo inferior a 0.2 grados. Por otro lado, el método tradicional con $\alpha = 0$ tiene un error medio mayor de 45 unidades (error máximo mayor que 101) y de más de 7 grados para la dirección (casi 15 grados el error máximo). Con $\alpha = 0.5$ los valores tienden a mejorar, siendo el error máximo del módulo mayor de 17 unidades y el de dirección casi de 2 grados.

También se representa en la tabla el error cometido en la localización subpixel del contorno. Este error viene medido en porcentaje relativo al tamaño del pixel. Es decir, un error igual a 50 representaría un error igual a la mitad de la longitud de un pixel. Se observa que los resultados de nuestro método son bastante buenos, ya que el error medio es inferior al 0.3% del tamaño del pixel, y el error máximo no llega al 0.6%.

El error cometido es diferente según la orientación de cada pixel borde. En los pixels donde la orientación es exactamente horizontal o vertical, los tres métodos obtienen una estimación bastante buena, pero para las orientaciones intermedias los errores varían en función de la orientación. Para poder ver mejor cómo es el

	Error del módulo				**Error de la dirección**				**Error de la posición**			
Método	Media	Desv.	Mín.	Max.	Media	Desv.	Mín.	Max.	Media	Desv.	Mín.	Max.
Tradic. $\alpha = 0$	45.7	34.4	0.0	101.8	7.1	3.7	0.0	14.7	–	–	–	–
Tradic. $\alpha = 0.5$	6.3	5.3	0.0	17.2	0.8	0.6	0.0	1.7	–	–	–	–
Prop. $\alpha = 0$	0.0	0.0	0.0	0.0	0.05	0.04	0.00	0.17	0.29	0.09	0.20	0.57
Prop. $\alpha = 0.5$	0.0	0.0	0.0	0.0	0.05	0.04	0.00	0.17	0.29	0.09	0.20	0.57

Tabla 2.3: Errores cometidos por cada método al obtener el contorno de una circunferencia de radio 20

error a medida que nos movemos por el borde, usemos la siguiente representación gráfica. En la figura 2.24 la circunferencia interior (marcada con un 0) representa los pixels del borde. Lo que se ha hecho es desplazar cada punto de la circunferencia en la dirección radial hacia el exterior una cantidad proporcional al error cometido para cada pixel en el cálculo del módulo (gráfica a) y de la dirección (gráfica b). El error se representa en valor absoluto, y por eso todas las curvas son exteriores a la circunferencia.

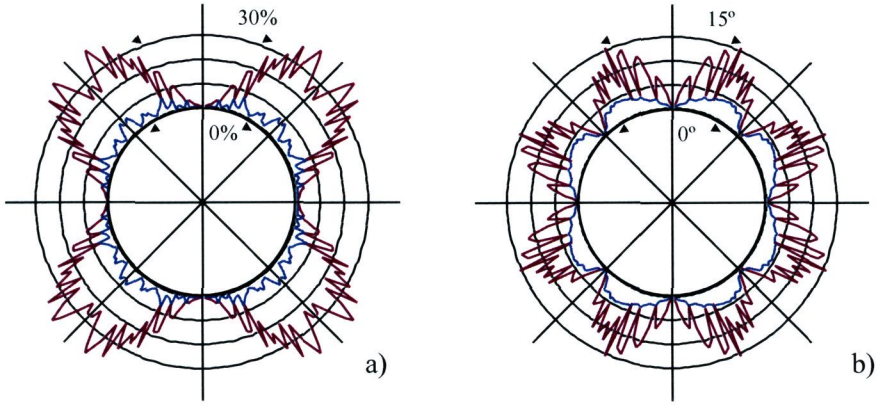


Figura 2.24: Representación circular del error en valor absoluto cometido sobre la imagen de un círculo de radio 40 en el cálculo del vector normal. a) error en el módulo. b) error en la orientación. La curva marrón representa el método tradicional con $\alpha = 0$. La curva azul representa el mismo método con $\alpha = 2 - \sqrt{2}$. La curva negra muestra los resultados de nuestro método.

Puede verse cómo nuestro método obtiene una estimación del módulo y de la orientación bastante precisa, mientras que el método tradicional comete errores en función de la orientación de cada pixel. Observando la gráfica correspondiente al error del módulo, vemos que en las 4 direcciones principales la estimación es bastante correcta, pero a medida que nos acercamos a los 45^o, el error aumenta llegando incluso a un 40% de error relativo. Usando $\alpha = 2 - \sqrt{2}$ se consigue un error menor, al tomar más pixels en consideración para el cálculo de las derivadas parciales. Aun así, el error en algunos pixels es cercano al 10%.

Para la gráfica de la orientación, los tres métodos afinan bastante bien en las 4 direcciones principales y en las 4 diagonales. Con alfa nulo llegamos a errores de hasta 15 grados, mientras que con α no nulo el error máximo baja a 5 grados aproximadamente.

2.7.3 Contorno recto semi-ideal

Probemos ahora con una imagen semi-ideal. Esto es, ha sido generada en el ordenador, impresa con una impresora láser, y adquirida luego con una cámara fotográfica, colocada en posición fronto-paralela con respecto a la página. En el centro de la figura 2.25 se observa la hoja de papel donde fue impresa la imagen generada. El objetivo es poder trabajar con una imagen adquirida por una cámara en lugar de trabajar directamente con la imagen sintética. De esta forma, como conocemos todos los detalles de la imagen (intensidades, orientaciones, posiciones) cuando fue generada, podremos compararlos con los obtenidos por nuestro método al ser aplicado sobre la imagen ya digitalizada. Seguramente habrán pequeños errores debido a la impresión en papel, ya que las intensidades generadas por la impresora no serán exactas, y debido luego al proceso de adquisición. Sin embargo, la idea es poder comprobar si nuestras hipótesis no están mal encaminadas, y producen resultados mejores que con el esquema tradicional.

La imagen generada representa un círculo dividido en 12 sectores de igual ángulo (30 grados), donde se van alternando las intensidades (50 el más oscuro, 100 y 200 el más claro). El interés de esta imagen radica por un

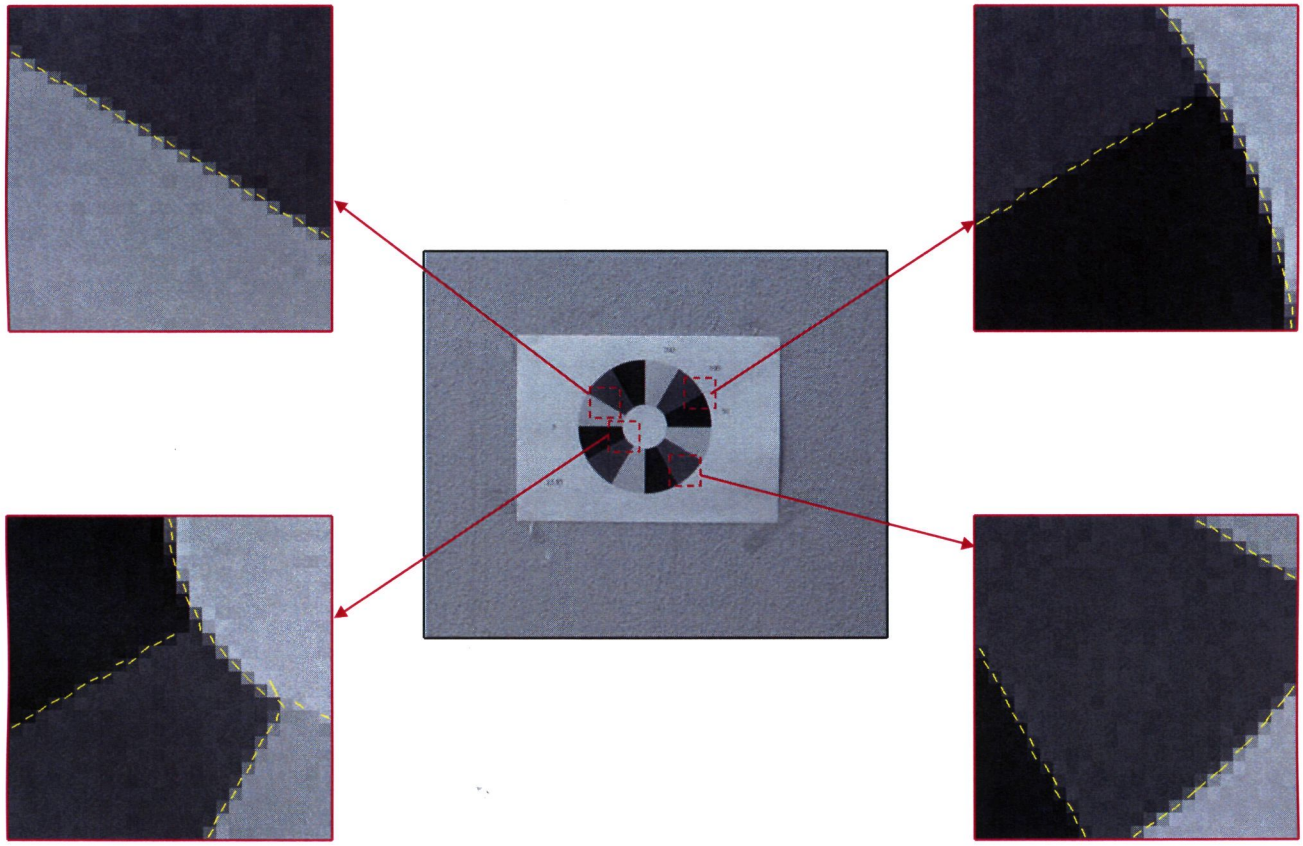


Figura 2.25: Contornos obtenidos por nuestro método para distintas zonas de la imagen que aparece en el centro.

lado en analizar el resultado de nuestra estimación para el módulo y por otro para la orientación. Realmente la orientación absoluta de cada sector no la sabemos, pero lo que sí sabemos es la orientación relativa entre ellos (30 grados) y eso es lo que vamos a comprobar.

Cálculo de la orientación Enumerando cada uno de los bordes entre sectores, de la misma manera que si fuesen las horas de un reloj, tendríamos desde el borde 1 hasta el 12. Para cada borde, vamos a calcular en primer lugar la pendiente media del contorno en todos los pixels etiquetados en cada borde. En la tabla 2.4 se muestra para cada borde cuál es ese valor para cada uno de los cuatro métodos, y cuál es la diferencia en grados con la pendiente del borde siguiente. Esto se hace así ya que la pendiente exacta al adquirir la imagen impresa no la conocemos con exactitud (sabemos que el borde número 1 tendrá una orientación cercana a 30^o, pero dependiendo de la orientación exacta al colocar la hoja de papel y al colocar la cámara no será una cantidad exacta, y de ahí el factor ε que aparece sumando al valor de pendiente ideal). Pero sí conocemos que la diferencia de pendiente entre sectores consecutivos ha de ser 30^o.

Como acabamos de mencionar, teóricamente el ángulo de cada sector debería ser exactamente 30 grados. Sea θ_i el ángulo estimado (columna 'Dif' de la tabla) según alguno de los métodos para el sector i. Si tomamos

	Valores ideales		MTAN		MTAP		MPAN		MPAP	
Num. Borde	PM	Dif.	PM	Dif.	PM	Dif.	PM	Dif.	PM	Dif.
1	$30+\varepsilon$	30	38.78	38.66	26.52	26.67	31.51	32.54	31.12	31.74
2	$60+\varepsilon$	30	53.76	14.98	64.34	37.82	58.76	27.25	59.50	28.38
3	$93+\varepsilon$	30	90.97	37.21	90.48	26.14	90.95	32.19	90.79	31.28
4	$120+\varepsilon$	30	127.09	36.12	116.21	25.73	119.77	28.82	120.11	29.32
5	$150+\varepsilon$	30	141.68	14.60	155.14	38.93	150.71	30.94	150.78	30.67
6	$180+\varepsilon$	30	180.83	39.15	180.41	25.27	180.75	30.04	180.64	29.86
7	$210+\varepsilon$	30	217.23	36.40	205.87	25.45	209.64	28.88	210.01	29.37
8	$240+\varepsilon$	30	234.52	17.29	245.26	39.39	243.67	34.04	242.62	32.61
9	$270+\varepsilon$	30	270.48	35.96	270.20	24.94	269.81	26.14	270.03	27.41
10	$300+\varepsilon$	30	308.72	38.24	296.73	26.53	302.75	32.94	301.75	31.72
11	$330+\varepsilon$	30	320.78	12.06	335.15	38.42	330.70	27.95	331.12	29.37
12	$360+\varepsilon$	30	360.12	39.34	359.85	24.70	358.98	28.27	359.39	28.26

Tabla 2.4: Errores cometidos por cada método al obtener la orientación del contorno de los 12 sectores de la imagen. MTAN/MTAP: método tradicional con alfa nulo / positivo. MPAN/MPAP: método propuesto con alfa nulo / positivo. PM: pendiente media.

	Mét. Tradic. $\alpha = 0$	**Mét. Tradic.** $\alpha = 0.59$	**Mét. Prop.** $\alpha = 0$	**Mét. Prop.** $\alpha = 0.59$
Error medio	10.18	5.76	2.11	1.34

Tabla 2.5: Error medio en grados, obtenido por cada método, al obtener la orientación del contorno de los 12 sectores de la imagen

una medida de error viendo las diferencias medias con respecto a 30 grados de la forma siguiente:

$$E_i = \frac{1}{12}\sum_{i=1}^{12}|\theta_i - 30|$$

obtenemos los resultados de la tabla 2.5.

Es decir, para $\alpha = 0$ el esquema tradicional tiene un error medio aproximado de 10 grados por sector, reduciéndolo a menos de 6 para $\alpha = 2 - \sqrt{2}$. Nuestro método tiene un error cercano a 2 grados con $\alpha = 0$, y usando $\alpha = 2 - \sqrt{2}$ el error desciende a 1.34 grados de media.

Nótese también cómo el error en el ángulo en el método tradicional es menor en las direcciones cercanas a las principales (3, 6, 9 y 12), pero es bastante mayor en las direcciones intermedias. En la figura 2.26 se ha representado un nuevo círculo de 12 sectores, donde se le ha asignado a cada sector un ángulo igual al estimado por los cuatro métodos para el borde de dicho sector con el siguiente. Vemos como no todos los sectores son igual de anchos en el método tradicional, mientras que usando nuestro método el resultado es bastante más homogéneo en todas las direcciones.

Cálculo del módulo Vayamos ahora con la estimación para el módulo del vector normal, el cual debería representar el salto de intensidad. Téoricamente se han usado 3 valores de intensidad diferentes (50, 100 y 200), pero después de los procesos de impresión y adquisición, y dependiendo también de las condiciones de luz en el momento de tomar la fotografía, la intensidad habrá variado bastante. Si calculamos el valor medio de cada una de las tres zonas, tomando un número considerable de pixels en el interior de cada sector para ello, obtenemos que los tres valores de intensidad medio de las tres zonas son 46.92, 94.09 y 158.94, en lugar de 50, 100 y 200 respectivamente.

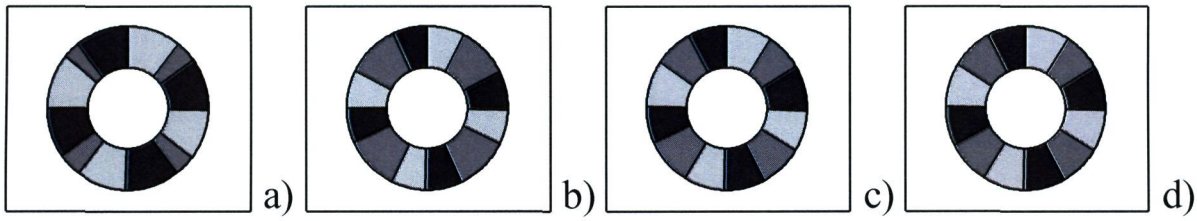


Figura 2.26: Error cometido en la diferencia en grados del ángulo obtenido para cada par de sectores consecutivos en la imagen de la figura anterior. a) Método tradicional $\alpha = 0$. b) Método tradicional $\alpha = 2 - \sqrt{2}$. c) Método propuesto $\alpha = 0$. d) Método propuesto $\alpha = 2 - \sqrt{2}$

	Salto medio	**MTAN**		**MTAP**		**MPAN**		**MPAP**	
Num. Borde		Mód.	Dif.	Mód.	Dif.	Mód.	Dif.	Mód.	Dif.
1	64.85	91.4	26.55	68.9	4.05	63.2	1.65	63.2	1.65
2	47.16	60.9	13.74	47.6	0.44	45.3	1.86	47.0	0.16
3	112.02	115.9	3.88	115.9	3.88	113.3	1.28	111.2	0.82
4	64.85	79.6	14.75	61.6	3.25	61.5	3.35	60.8	4.05
5	47.16	69.7	22.54	52.4	5.24	47.0	0.16	46.5	0.66
6	112.02	125.2	13.18	125.2	13.18	107.7	4.32	106.3	5.72
7	64.85	86.2	21.35	66.0	1.15	63.0	1.85	63.0	1.85
8	47.16	62.1	14.94	48.8	1.64	46.9	0.26	46.7	0.46
9	112.02	109.7	2.32	109.6	2.42	109.0	3.02	108.6	3.42
10	64.85	90.4	25.55	68.2	3.35	64.1	0.75	64.2	0.65
11	47.16	73.3	26.14	54.2	7.04	45.6	1.56	47.1	0.06
12	112.02	131.5	19.48	131.4	19.38	112.3	0.28	112.7	0.68

Tabla 2.6: Errores cometidos por cada método al obtener el salto de intensidad a ambos lados del contorno de los 12 sectores de la imagen. MTAN/MTAP: método tradicional con alfa nulo / positivo. MPAN/MPAP: método propuesto con alfa nulo / positivo.

Por lo tanto, podemos estimar qué diferencia de intensidad media debería aparecer en cada borde entre sectores, y comparar entre sí los métodos. En la tabla 2.6 se muestran los resultados de evaluar el módulo del vector normal para los pixels borde de cada uno de los 12 contornos, y la diferencia entre este valor y el estimado para el salto de intensidad medio a partir de los valores medidos en la imagen en el interior de cada sector.

De nuevo podemos dar una medida del error medio cometido por cada método, atendiendo a las diferencia entre el valor obtenido para el módulo del vector en cada borde (columna 'Mod.'), y el salto medio que se supone había a ambos lados (columna 'Salto medio'). Sea D_i esta diferencia de intensidad, medida para el borde i por alguno de los métodos. Calculando el error medio como

$$E_i = \frac{1}{12} \sum_{i=1}^{12} D_i$$

obtenemos los resultados de la tabla 2.7.

Es decir, para $\alpha = 0$ el esquema tradicional tiene un error medio aproximado de 17 unidades en cada borde, reduciéndolo a menos de 6 para $\alpha = 2 - \sqrt{2}$. Nuestro método tiene un error inferior a 2 unidades para cualquier valor de α.

	Mét. Tradic. $\alpha = 0$	Mét. Tradic. $\alpha = 0.59$	Mét. Prop. $\alpha = 0$	Mét. Prop. $\alpha = 0.59$
Error medio	17.04	5.42	1.70	1.68

Tabla 2.7: Error medio en unidades de intensidad obtenido por cada método, al obtener el salto de intensidad a ambos lados del contorno de los 12 sectores de la imagen.

2.7.4 Imagen real digitalizada

Finalmente, hemos aplicado nuestro esquema a la foto de la figura 2.27, donde pueden apreciarse diferentes zonas que han sido agrandadas para visualizar mejor el resultado de nuestro método. Puede verse como visualmente los contornos parecen tener orientaciones y posiciones sub-pixel bastante acertadas, al menos en las zonas donde se cumplen las tres condiciones que mencionábamos en la sección donde se comentó el algoritmo (sección 2.6).

Puede verse también que, incluso el contorno del botón de la camisa del Popeye, proporciona resultados bastante buenos, teniendo en cuenta que dicho botón tiene forma circular de unos 8 pixels de diámetro aproximadamente.

2.8 Conclusiones

Se ha desarrollado un esquema capaz de encontrar en imágenes ideales sintéticas la orientación y localización sub-pixel de un contorno recto de forma exacta, así como el salto de intensidad producido a ambos lados del borde. Posteriormente, y a partir de dicho esquema, hemos desarrollado un método para localizar los contornos sobre una imagen real, asumiendo que localmente dichos contornos se comportan como líneas rectas, y hemos visto varios ejemplos prácticos de aplicación.

Sin embargo, para ser rigurosos, si el contorno no fuese exactamente recto, se estaría cometiendo un pequeño error incluso en las imágenes ideales, tal y como se vio en la sección 2.7.2. En el próximo capítulo desarrollaremos un nuevo esquema que nos permitirá mejorar este cálculo, al aproximar los contornos por curvas de grado dos en lugar de líneas rectas como hemos hecho ahora. Ello nos permitirá obtener además otro parámetro característico del contorno que pasa por un pixel: la curvatura.

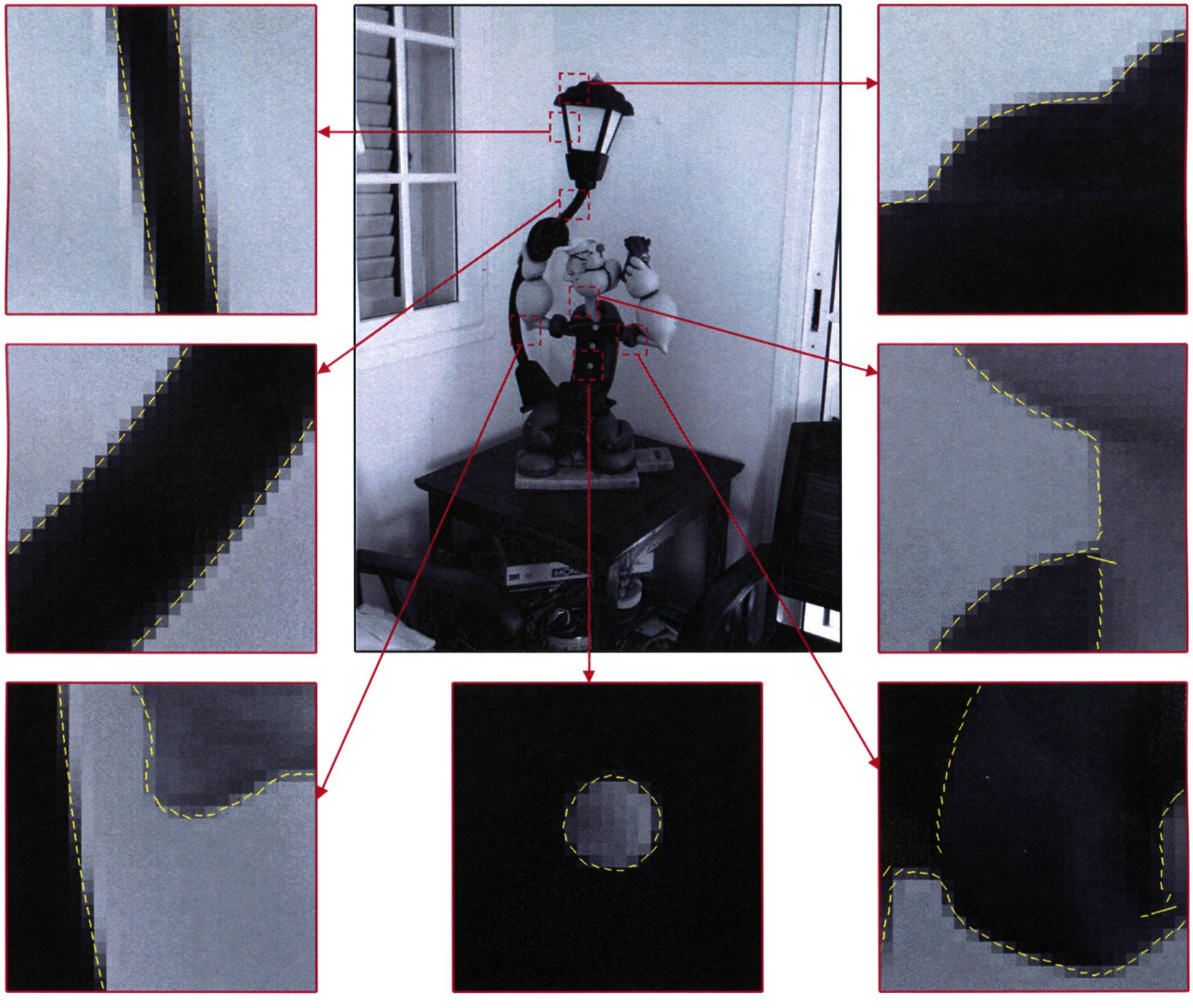


Figura 2.27: Contornos detectados por nuestro método en la foto de una lámpara Popeye.

Capítulo 3

Contornos de segundo orden

- Si en lugar de buscar rectas en el interior del pixel buscáramos curvas, podríamos estimar también la curvatura en los bordes.
- De hecho, disponemos de tres columnas alrededor de cada pixel, con lo cual tienes tres grados de libertad....

JUNIO 2001

Acabamos de ver en el capítulo anterior un método que nos calcula la posición y orientación del contorno que pasa por un pixel de forma exacta (al menos en imágenes ideales en el sentido de nuestras hipótesis de partida). Pero existe otro parámetro característico del contorno que también podría ser interesante estimar: la curvatura. Este parámetro nos informa de cuan recto es el contorno en una cierta posición, y en caso de ser curvo, hacia qué lado es la desviación. Realmente lo que mide es el ritmo de cambio de la pendiente del contorno con respecto a la distancia sobre la curva [GOE70].

Por ejemplo, en la figura 3.1, la curvatura (en valor absoluto) en los puntos P_i viene dada por el valor

$$|K(P_i)| = \frac{1}{R_i}$$

donde R_i es conocido como el **radio de curvatura**. Esto es, el radio de la circunferencia que es tangente a la curva en dicho punto y que mejor aproxima a la curva en un entorno de dicho punto. A mayor curvatura, menor será el radio de la circunferencia tangente, y viceversa.

En este capítulo analizaremos primero brevemente la forma tradicional de calcular la curvatura de un contorno en una imagen, y posteriormente propondremos un nuevo método que nos permitirá calcular la localización, orientación, y curvatura del contorno en un pixel en un mismo esquema. Finalmente se verán algunos ejemplos de aplicar dicho método a varias imágenes sintéticas y reales.

3.1 Cálculo tradicional de curvaturas en imágenes 2D

Aunque hay muchas formas de abordar el problema de estimar la curvatura, la gran mayoría cae dentro de una de estas categorías:

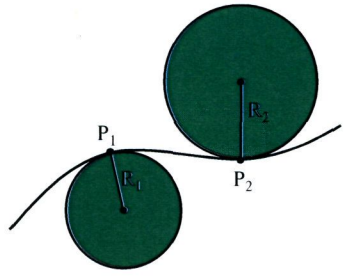


Figura 3.1: Radios de curvatura en dos puntos de una curva

3.1.1 A partir de las segundas derivadas de la imagen

Ya hemos visto cómo la información de la orientación de una curva viene dada por su primera derivada, y es por eso que se utilizaba el vector gradiente para obtenerla. Ya que la curvatura está relacionada con las segundas derivadas, el método usual es calcular éstas para a partir de ellas obtener el valor de la curvatura [DAN01].

Veamos primero el caso continuo. Sea $f(x, y)$ nuestra imagen de entrada (anterior a la discretización), tal que $f(x, y)$ es continua y derivable en todo su dominio[1]. El vector gradiente en un punto $P = (x_0, y_0)$ viene dado por la expresión

$$\nabla f(x_0, y_0) = [f_x(x_0, y_0), f_y(x_0, y_0)] \tag{3.1}$$

y se cumple que es un vector normal a la curva de nivel

$$f(x, y) = f(x_0, y_0)$$

en dicho punto. A partir de dicho vector se obtiene la pendiente exacta de la curva en P.

Para calcular la curvatura de la curva de nivel en el punto P, utilizamos la expresión

$$k(x_0, y_0) = \frac{f_x^2(x_0, y_0) f_{yy}(x_0, y_0) + f_y^2(x_0, y_0) f_{xx}(x_0, y_0) - 2 f_x(x_0, y_0) f_{xy}(x_0, y_0) f_y(x_0, y_0)}{\|\nabla f(x_0, y_0)\|^3} \tag{3.2}$$

definida en la literatura como **función curvatura** [APO66].

Sin embargo, esta expresión es solamente para el caso continuo. Para poder dar entonces una estimación en el caso discreto, se hace necesario encontrar máscaras de convolución a partir de las discretizaciones obtenidas para las segundas derivadas f_{xx}, f_{yy} y f_{xy}, al igual que hicimos con las primeras derivadas f_x y f_y en el capítulo inicial. Una vez obtenidas, se evalúan todas ellas en el pixel correspondiente, y se calcula el valor estimado para $k(x_0, y_0)$.

El problema que existe es que la curvatura es un valor extremadamente sensible, con lo cual variaciones mínimas de la intensidad en los pixels producen grandes variaciones en el valor de la curvatura. Es por ello que no suele utilizarse en muchos casos, al menos con la imagen sin preproceso de ninguna clase.

3.1.2 A partir del gradiente de los pixels vecinos

Otro método que suele usarse consiste en utilizar la información de pixels vecinos en la curva [BEN75, GRO78, KIT82]. Hay que tener en cuenta que normalmente cuando buscamos los pixels por donde pasa el borde, estos suelen formar una secuencia conectada de pixels, como se ve en la figura 3.2. Si dicha secuencia ha sido bien obtenida, debería formar un conjunto de pixels conectados entre sí, de tal forma que cada pixel tuviera

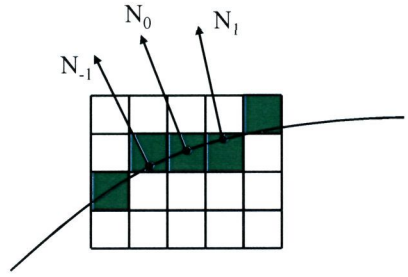


Figura 3.2: A partir de las direcciones de los vectores normales de 3 pixels consecutivos sobre el borde, puede darse una estimación de la curvatura para el pixel central.

exclusivamente dos vecinos conectados en el conjunto, uno a cada lado. Es lo que se conoce como una línea de grosor 1 pixel, tal y como la que produce nuestro método del capítulo previo.

En ese caso, ya que conocemos los vectores normales a la curva en cada uno de esos pixels, podemos dar una estimación de la curvatura para cada pixel basándonos en el cambio de orientación producido entre su vector normal y el de sus vecinos. Sin embargo, éste también es un método que dista mucho de ser exacto, ya que habría que conocer con exactitud sobre qué punto exacto del contorno (en el plano continuo, no en los pixels) está situado cada uno de los vectores normales, ya que para medir la variación de la pendiente entre los puntos hay que conocer a qué distancia se encuentran a lo largo de la curva. Teniendo en cuenta que la mayoría de los métodos trabajan con precisión pixel y no sub-pixel, este cálculo sería bastante complicado de realizar con exactitud.

3.1.3 A partir de la expresión de una función que aproxime a la imagen

Una tercera forma diferente de obtener la curvatura es hacerlo analíticamente a partir de la expresión de una cierta función $p(x, y)$, cuya discretización $P(x_i, y_j)$ es la que mejor aproxima a las intensidades de los pixels en una cierta vecindad alrededor del pixel seleccionado [TSA94, LEE93]. Es decir, de nuevo vuelve a suponerse que la imagen de entrada (anterior a la discretización), $f(x, y)$, es continua y derivable en todo su dominio, y lo que se busca es encontrar la función $p(x, y)$ dada por

$$p(x,y) = k_0 + k_1 x + k_2 y + k_3 x^2 + k_4 xy + k_5 y^2 + k_6 x^3 + k_7 x^2 y + k_8 xy^2 + k_9 y^3$$

que mejor la aproxime en un cierto entorno, considerando un sistema de referencia centrado en el centro geométrico del pixel que estamos estudiando.

Para ello, y haciendo uso del mismo modelo de adquisición expuesto en el capítulo 1 (ecuación 1.6, se considera que la función discretizada adquirida en la imagen viene dada por la ecuación

$$P(x_i, y_j) = \frac{1}{h^2} \int_{y_j - h/2}^{y_j + h/2} \int_{x_i - h/2}^{x_i + h/2} p(x,y) dx dy$$

donde h es la longitud de un pixel. Resolviendo la integral obtenemos

$$\begin{aligned} P(x_i, y_j) \;=\; & k_0 + k_1 x_i + k_2 y_j + k_3 \left(x_i^2 + \frac{1}{12} \right) + k_4 x_i y_j + k_5 \left(y_j^2 + \frac{1}{12} \right) + \\ & + k_6 x_i \left(x_i^2 + \frac{1}{4} \right) + k_7 y_j \left(x_i^2 + \frac{1}{12} \right) + k_8 x_i \left(y_j^2 + \frac{1}{12} \right) + k_9 y_j \left(y_j^2 + \frac{1}{4} \right) \end{aligned}$$

[1] Recordemos que esa es la hipótesis inicial usada en la mayoría de los trabajos, pero que no se usa en el nuestro.

A continuación, se minimiza el error cuadrático medio cometido por aproximar esta función a una cierta vecindad del pixel central, el cual viene dado por la expresión

$$e(k_0, k_1, ..., k_9) = \sum_i \sum_j (P(x_i, y_j) - F(i,j))^2$$

En este caso la sumatoria indica la extensión de la vecindad considerada. Como resultado de la minimización se obtienen los valores óptimos para los coeficientes k_i de la función $p(x,y)$. Finalmente, se evalúan los parámetros del contorno (gradiente y curvatura) a partir de las expresiones 3.1 y 3.2 respectivamente, pero utilizando la función $p(x,y)$ en lugar de $f(x,y)$, y evaluándola en centro geométrico del pixel en cuestión, es decir, en el punto $(0,0)$.

Los resultados obtenidos con este método suelen ser bastante satisfactorios, sobre todo en zonas donde la hipótesis de que la función $f(x,y)$ es continua y derivable parece cumplirse. Éste es el caso de zonas con intensidad homogéneas, como degradados, o zonas con bordes ligeramente difuminados. Sin embargo, no determinan la localización sub-pixel del contorno, y no son exactos en imágenes ideales en el sentido de nuestras hipótesis de partida ($f(x,y)$ es discontinua en los bordes).

Por otro lado, al evaluar los parámetros del borde en el punto $(0,0)$, lo que están haciendo realmente es interpretar el contorno como la curva de nivel de $p(x,y)$ que pasa por el centro del pixel. Pero curvas de nivel que atraviesen el pixel seleccionado hay infinitas, y cada una de ellas tiene infinitos puntos que la forman dentro del pixel. Y cada uno de esos puntos tendrá un valor de gradiente y curvatura diferente. Es cierto que serán valores bastante similares, pero no son exactos en un sentido riguroso.

3.1.4 Otras alternativas

Además de las tres formas mencionadas existen algunas otras, tal y como se menciona en [NIX02]: aproximando el contorno por pequeños tramos de curva continua (generalmente polinomios cúbicos) y evaluando la ecuación 3.2 en ellos, definiendo la curvatura como una función de los cambios en la intensidad de la imagen, o aproximándola en función de la autocorrelación medida al desplazar una cierta ventana de pixels en las cuatro direcciones principales. Esta última técnica suele dar buenos resultados para la detección de esquinas en la imagen, como se menciona en [HAR88].

Sin embargo, al carecer de una información precisa a nivel subpixel de la localización de la curva del contorno, todas ellas producen un cierto error en la estimación. Vamos por tanto ahora a desarrollar un método que sí nos permita obtener la curvatura exacta en imágenes ideales. Pero antes, veamos otra forma alternativa para nuestro método propuesto en el capítulo previo para el cálculo de la pendiente y de la localización del contorno. Esta otra alternativa es necesaria ya que partiendo de este método para contornos de primer orden es difícil generalizarlo para desarrollar el de segundo orden. Así que necesitamos usar otra técnica que sí nos permita realizar esta generalización.

3.2 Método simplificado de primer orden

Cuando comenzamos el desarrollo del método (sección 2.1.2), partimos de que $F_{i,j}$ era una imagen compuesta por un único contorno recto que separaba dos zonas de intensidades A y B, donde el valor de los pixels se obtenía de acuerdo a nuestra hipótesis inicial. Dicho contorno recto atravesaba el pixel central de la imagen y tenía una pendiente entre 0 y 45° (ver figura 3.3). Es lo que llamamos imagen ideal con un contorno de primer orden (expresión 2.1).

En un caso real, el contorno no va a ser exactamente recto, con lo cual suponer que lo es no es más que una aproximación que asumimos en la vecindad del pixel, tal y como se discutió en la sección 2.6.

Formalmente, lo que estamos haciendo puede definirse de la siguiente manera: situemos primero el origen de coordenadas en el centro geométrico del pixel central, y consideremos los ejes x e y como se ven en la figura

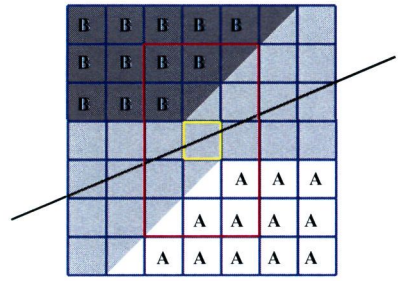


Figura 3.3: Los pixels con color gris pueden tomar valores intermedios entre A y B en función de la posición y orientación exacta del contorno.

3.4. Nótese que esta vez hemos tomado la dirección de las y positivas hacia arriba, al contrario de como están direccionados los pixels en la imagen $F_{i,j}$, simplemente por seguir la forma habitual de representar funciones en el plano.

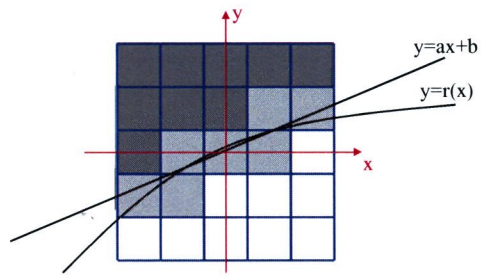


Figura 3.4: Curva del contorno y aproximación lineal usada por nuestro método.

Sea $y = r(x)$ la curva que representa el contorno, cuya expresión exacta desconocemos. Lo que sí sabemos es que la pendiente de la curva en $x = 0$ está entre 0 y 1. Una aproximación lineal a la curva alrededor del punto $x = 0$ puede obtenerse como $y = a + bx$, al estilo de cuando se hace una aproximación a través del polinomio de Taylor de grado 1 de una función

$$P_1(x) = r(0) + r'(0)x = a + bx$$

Realmente no hacemos como Taylor, pues éste obtiene como aproximación la recta cuya posición y primera derivada coincide con la curva del contorno. En nuestro caso, la recta que buscamos es la que mejor se ajusta a los valores de intensidad de los pixels adyacentes, y no es necesariamente la misma que la de Taylor. Por lo tanto, lo que nuestro método trata de hacer puede interpretarse como encontrar **los valores de a y b que mejor se ajustan a las intensidades de los pixels** en un cierto entorno centrado en el pixel central. Una vez hayamos localizado los coeficientes de la recta, la posición de ésta puede interpretarse como la posición del contorno dentro del pixel, y la pendiente de la recta considerarse como la derivada de la curva del contorno sobre la vertical central del pixel. Por lo tanto, el vector normal a dicha recta será una buena aproximación del vector normal al contorno en dicho punto.

El entorno utilizado para realizar los cálculos tenía diferente forma según el valor de α usado en las máscaras (ver figuras 2.20 y 2.21), e incluso, según el valor que se estuviera calculando en cada momento (S_x, S_y o r_y), era también diferente. Para simplificar todos estos entornos diferentes, tomemos ahora una única ventana, y

calculemos con los valores de los pixels interiores los coeficientes a y b, a partir de los cuales podremos estimar la orientación y localización exacta de la recta que estamos buscando. Tomemos por ejemplo un entorno de 5×3, centrado en el pixel central (ver figura 3.3). Es básicamente el mismo que se usaba para el caso $\alpha = 0$ (figura 2.20) pero obviando los pixels más alejados de la columna central, para que la forma final del entorno salga rectangular. Hay que recordar también que estamos en el caso para pendientes entre 0 y 1, y de ahí que la ventana sea más larga en la dirección y que en la x. Para contornos con pendiente mayor que 1, la ventana sería lógicamente de 3×5.

Por otro lado, vamos también a trabajar directamente con los valores de la imagen de entrada, $F_{i,j}$, sin tener que usar los valores de las imágenes de parciales más que para detectar los máximos locales y el octante en el que nos encontramos. De esta forma, en las expresiones que usemos para detectar los parámetros del contorno sólo aparecerán los valores de F y no los de F_x y F_y.

3.2.1 Cálculo de los coeficientes de la recta

Tomemos por tanto dicho entorno, centrado en el pixel central (i, j). Dentro de esta ventana, asumimos que el contorno se comporta como una línea recta $y = a + bx$. Eso significa que, según nuestra hipótesis inicial, la intensidad de cada pixel viene dada por la siguiente expresión:

$$F(i,j) = \frac{P_{i,j}}{h^2} A + \frac{h^2 - P_{i,j}}{h^2} B = B + \frac{A-B}{h^2} P_{i,j}$$

donde en este caso $P_{i,j}$ es el área interior al pixel (i, j) que se encuentra bajo la recta. Por tanto, $0 \leq P_{i,j} \leq h^2$.

Tomemos cualquiera de las tres columnas que forman la ventana, y sumemos los valores de los 5 pixels que comprenden. En ese caso, los valores obtenidos deberían corresponder a las siguientes expresiónes:

$$\begin{aligned}
S_L &= \sum_{n=j-2}^{j+2} F_{i-1,n} = 5B + \frac{A-B}{h^2} \sum_{n=j-2}^{j+2} P_{i-1,n} = 5B + \frac{A-B}{h^2} L \\
S_M &= \sum_{n=j-2}^{j+2} F_{i,n} = 5B + \frac{A-B}{h^2} \sum_{n=j-2}^{j+2} P_{i,n} = 5B + \frac{A-B}{h^2} M \\
S_R &= \sum_{n=j-2}^{j+2} F_{i+1,n} = 5B + \frac{A-B}{h^2} \sum_{n=j-2}^{j+2} P_{i+1,n} = 5B + \frac{A-B}{h^2} R
\end{aligned} \tag{3.3}$$

donde L, M y R indican el área interior a cada columna que se encuentra bajo la recta, tal y como se ve en la figura 3.5.

Tratándose de una línea recta que atraviesa el pixel central, con pendiente entre 0 y 1, deducimos que dicha recta siempre atravesará los límites de nuestra ventana de izquierda a derecha, y nunca cortará el borde superior e inferior de la ventana. Esto significa que dichas áreas pueden ser calculadas analíticamente de forma sencilla, mediante una integral iterada, de la siguiente manera:

$$\begin{aligned}
L &= \int_{-3h/2}^{-h/2} (a + bx + 5h/2)\, dx = ah - bh^2 + \frac{5}{2}h^2 \\
M &= \int_{-h/2}^{h/2} (a + bx + 5h/2)\, dx = ah + \frac{5}{2}h^2 \\
R &= \int_{h/2}^{3h/2} (a + bx + 5h/2)\, dx = ah + bh^2 + \frac{5}{2}h^2
\end{aligned}$$

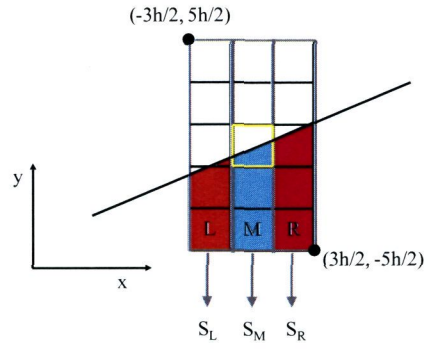


Figura 3.5: La suma de intensidades de cada columna depende del área interior a la columna bajo la recta.

Vemos que la segunda ecuación correspondiente a M sólo depende de a y no de b, con lo cual podemos usarla para hallar una expresión para a, de esta manera:

$$\begin{aligned} S_M &= 5B + \frac{A-B}{h^2}M = 5B + \frac{A-B}{h^2}\left(ah + \frac{5}{2}h^2\right) \Rightarrow \\ &\Rightarrow \quad a = \frac{2S_M - 5\,(A+B)}{2(A-B)}h \end{aligned}$$

Una vez conocido a, podría hallarse b utilizando cualquiera de las columnas izquierda o derecha, pero para que el esquema sea más simétrico y poder así usar información de ambas columnas simultáneamente, usaremos la resta de ambas expresiones, ya que el término donde se encuentra b tiene el signo cambiado. De esta manera:

$$\begin{aligned} S_R - S_L &= \frac{A-B}{h^2}\,(R-L) = \frac{A-B}{h^2}\,(2bh^2) \Rightarrow \\ &\Rightarrow \quad b = \frac{S_R - S_L}{2\,(A-B)} \end{aligned}$$

Como los valores de S_L, S_M y S_R son conocidos, ya que se calculan a partir de los valores de los pixels, nos faltaría conocer los valores de A, B y h. Para h, su valor exacto es irrelevante, con lo cual lo usual es darle el valor $h = 1$. Para A y B, y ciñéndonos al interior de la ventana que estamos utilizando (ver figura 3.3), podemos encontrar estimaciones fácilmente usando los pixels que sabemos con seguridad que no tocan a la recta. De esta forma, con los tres pixels de la esquina superior izquierda estimaríamos el valor de A, y con los tres de la esquina inferior derecha el valor de B. Así nos quedaría:

$$\begin{aligned} A &= \frac{F_{i,j+2} + F_{i+1,j+2} + F_{i+1,j+1}}{3} \\ B &= \frac{F_{i-1,j-1} + F_{i-1,j-2} + F_{i,j-2}}{3} \end{aligned} \tag{3.4}$$

Con esto concluimos que, usando una ventana de 5×3 centrada en el pixel, podemos obtener de forma sencilla la expresión para la recta que mejor aproxima la curva del contorno. Ahora usaremos la expresión de esta recta para obtener los datos que queríamos, que son: posición de la recta en $x = 0$ y vector normal a la recta. El salto de intensidad a ambos lados del contorno se calcula directo, ya que es $A - B$.

3.2.2 Cálculo de los parámetros del contorno

Ya que nuestro contorno sigue la expresión $y = a + bx$, tal y como se ve en la figura 3.6, calcular la posición exacta del borde es inmediato. Éste cortará a la vertical central del pixel en el punto $(0, d)$, donde

$$d = y(0) = a$$

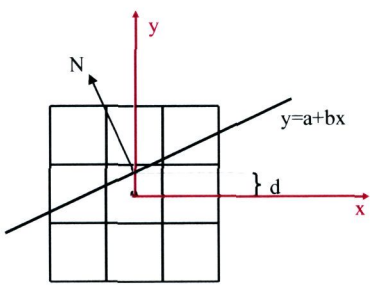


Figura 3.6:

Calcular el vector normal también es sencillo. Tenemos que el vector normal a la recta es

$$N = [b, -1]$$

Como queremos que su módulo sea igual a $A - B$, entonces

$$N = \frac{A - B}{\sqrt{1 + b^2}} [b, -1]$$

Con esto llegamos al siguiente lema:

Lema 3.1 *Sea $F_{i,j}$ una imagen por cuyo pixel (i, j) pasa un contorno con pendiente entre 0 y 1, que separa dos zonas de intensidad A (debajo del contorno) y B (encima del contorno). Si suponemos que en la vecindad de ese pixel, el borde se comporta como una línea recta, podemos estimar los parámetros de dicho borde (posición, pendiente y magnitud del salto de intensidad a ambos lados del contorno) calculando previamente la recta que mejor se aproxima a dicho borde. Los pasos son los siguientes:*

a) obtenemos las sumas de las columnas izquierda, central y derecha, de la siguiente manera:

$$S_L = \sum_{n=j-2}^{j+2} F_{i-1,n}$$

$$S_M = \sum_{n=j-2}^{j+2} F_{i,n}$$

$$S_R = \sum_{n=j-2}^{j+2} F_{i+1,n}$$

b) obtenemos también una estimación para los valores de intensidad a ambos lados del borde

$$A = \frac{F_{i,j+2} + F_{i+1,j+2} + F_{i+1,j+1}}{3}$$

$$B = \frac{F_{i-1,j-1} + F_{i-1,j-2} + F_{i,j-2}}{3}$$

c) a continuación detectamos los coeficientes de la recta $y = a + bx$ como sigue:

$$\begin{aligned} a &= \frac{2S_M - 5\,(A+B)}{2(A-B)} \\ b &= \frac{S_R - S_L}{2\,(A-B)} \end{aligned}$$

d) finalmente calculamos los parámetros de la recta

$$\begin{aligned} \text{corte con la vertical central del pixel} &: (0, a) \\ \text{vector normal} &: N = \frac{A-B}{\sqrt{1+b^2}}\,[b, -1] \end{aligned}$$

3.3 Método simplificado de segundo orden

Para desarrollar un método que permita estimar la curvatura del contorno en un pixel a partir del método anterior, simplemente habrá que cambiar de una aproximación lineal a una cuadrática. Esto es lógico ya que, si realmente estamos reconociendo que la línea del contorno no es recta sino que puede tener una ligera curvatura, entonces no tiene sentido aproximar dicha línea por una recta. En su lugar habrá que aproximar por una parábola. Es decir, volviendo al polinomio de Taylor, pero esta vez de grado 2, de una función $r(x)$ alrededor del punto $x = 0$, éste tiene la expresión:

$$P_2(x) = r(0) + r'(0)x + \frac{r''(0)}{2}x^2 = a + bx + cx^2 \tag{3.5}$$

Por lo tanto, lo que haremos en esta ocasión será tratar de encontrar los coeficientes de la parábola que mejor se ajustan a los valores de intensidad de los pixels cercanos al pixel en cuestión, y obtener los valores de posición, pendiente y curvatura de dicha parábola, los cuales serán las estimaciones que daremos para los parámetros del contorno.

Así que procedamos de la misma forma que hicimos antes. La situación será la que se muestra en la figura 3.7. Vamos a tratar de encontrar la parábola que pasa por el pixel central. Usaremos una ventana de iguales dimensiones que la de antes, 5×3. Sin embargo, en este caso aparece una diferencia con respecto al método lineal. Ahora no podemos garantizar que la parábola cruce la ventana de izquierda a derecha. Ya volveremos a este detalle más adelante. Por ahora, asumamos que la parábola no corta los límites inferior y superior de la ventana.

3.3.1 Cálculo de los coeficientes de la parábola

Si realizamos las sumas de los valores de los pixels en las tres columnas, encontramos idénticas expresiones que en el caso lineal (ecuaciones 3.3). La diferencia radica ahora en las expresiones para las áreas L, M y R. Atendiendo a la figura 3.8, dichas expresiones son ahora las siguientes:

$$\begin{aligned} L &= \int_{-3h/2}^{-h/2} \left(a + bx + cx^2 + 5h/2\right) dx = ah - bh^2 + \frac{13}{12}ch^3 + \frac{5}{2}h^2 \qquad (3.6) \\ M &= \int_{-h/2}^{h/2} \left(a + bx + cx^2 + 5h/2\right) dx = ah + \frac{1}{12}ch^3 + \frac{5}{2}h^2 \\ R &= \int_{h/2}^{3h/2} \left(a + bx + cx^2 + 5h/2\right) dx = ah + bh^2 + \frac{13}{12}ch^3 + \frac{5}{2}h^2 \end{aligned}$$

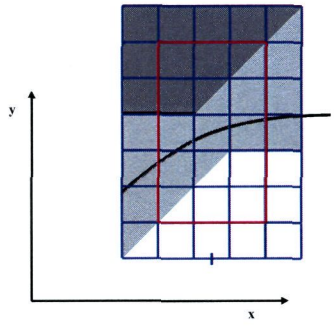


Figura 3.7: Se asume que la parábola cruza la ventana de izquierda a derecha. Los pixels en gris claro son los que pueden tomar valores intermedios entre A y B.

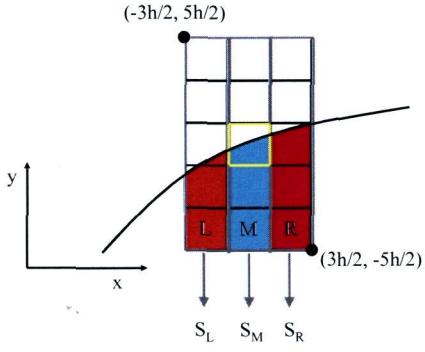


Figura 3.8:

En esta ocasión, tenemos un sistema lineal de tres ecuaciones y tres incógnitas, (a, b, c), fácil de resolver, cuya solución es la siguiente:

$$\begin{aligned} a &= \frac{26S_M - S_L - S_R}{24\,(A-B)}h + \frac{(5-125h)A - (5+115h)B}{48\,(A-B)} \\ b &= \frac{S_R - S_L}{2\,(A-B)} + \frac{5\,(h-1)}{4h} \\ c &= \frac{S_L + S_R - 2S_M}{2h\,(A-B)} + \frac{5\,(h-1)}{4h^2} \end{aligned}$$

Los valores de A y B se estiman igual que en el caso lineal (ecuación 3.4) y a h se le asigna valor 1, quedando expresiones más simples.

Con esto concluimos que, de forma similar al caso anterior, usando una ventana de 5×3 centrada en el pixel, podemos obtener de forma sencilla la expresión para la parábola que mejor aproxima la curva del contorno (aunque recordemos que hasta el momento hemos fijado la restricción de que la parábola cruza la ventana de izquierda a derecha). Ahora usaremos la expresión de esta parábola para obtener los datos que queríamos, que son: posición de la curva en $x = 0$, vector normal a la curva en ese punto, y curvatura en ese punto. El salto de intensidad a ambos lados del contorno se calcula directo, ya que es $A - B$.

3.3.2 Cálculo de los parámetros del contorno

Ya que nuestro contorno sigue la expresión $y = a+bx+cx^2$, tal y como se ve en la figura 3.9, calcular la posición exacta del borde es inmediato:

$$d = y(0) = a$$

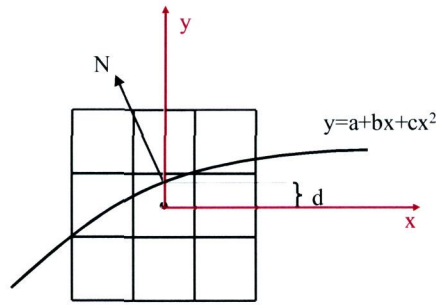


Figura 3.9:

El vector normal, a diferencia del caso lineal, será diferente en cada punto de la curva. Estamos interesados en el punto $x = 0$. Por tanto

$$\begin{aligned} N(x) &= [b + 2cx, -1] \\ N(0) &= [b, -1] \end{aligned}$$

Como queremos que su módulo sea igual a $A - B$, entonces

$$N = \frac{A-B}{\sqrt{1+b^2}}[b, -1] \tag{3.7}$$

Finalmente, para calcular la curvatura, sabemos que

$$K(x) = \frac{y''}{(1+(y')^2)^{3/2}} = \frac{2c}{(1+(b+2cx)^2)^{3/2}}$$

y en $x = 0$ obtenemos que

$$K = \frac{2c}{(1+b^2)^{3/2}} \tag{3.8}$$

Con esto llegamos al siguiente lema:

Lema 3.2 *Sea $F(i,j)$ una imagen por cuyo pixel (i,j) pasa un contorno con pendiente entre 0 y 1, que separa dos zonas de intensidad A (debajo del contorno) y B (encima del contorno). Si suponemos que en la vecindad de ese pixel, el borde se comporta como una parábola, podemos estimar los parámetros de dicho borde (posición, pendiente, magnitud del salto de intensidad y curvatura, medidos sobre la vertical central del pixel), calculando previamente la parábola que mejor se aproxima a dicho borde. Los pasos son los siguientes:*

a) obtenemos las sumas de las columnas izquierda, central y derecha, de la siguiente manera:

$$\begin{aligned} S_L &= \sum_{n=j-2}^{j+2} F_{i-1,n} \\ S_M &= \sum_{n=j-2}^{j+2} F_{i,n} \\ S_R &= \sum_{n=j-2}^{j+2} F_{i+1,n} \end{aligned}$$

b) obtenemos también una estimación para los valores de intensidad a ambos lados del borde

$$\begin{aligned} A &= \frac{F_{i,j+2} + F_{i+1,j+2} + F_{i+1,j+1}}{3} \\ B &= \frac{F_{i-1,j-1} + F_{i-1,j-2} + F_{i,j-2}}{3} \end{aligned}$$

c) a continuación detectamos los coeficientes de la parábola $y = a + bx + cx^2$ como sigue:

$$\begin{aligned} a &= \frac{26S_M - S_L - S_R - 60(A+B)}{24(A-B)} \\ b &= \frac{S_R - S_L}{2(A-B)} \\ c &= \frac{S_L + S_R - 2S_M}{2(A-B)} \end{aligned}$$

d) finalmente calculamos los parámetros de la parábola

$$\begin{aligned} \text{corte con la vertical central del pixel} &: (0, a) \\ \text{vector normal} &: N = \frac{A-B}{\sqrt{1+b^2}}[b, -1] \\ \text{curvatura} &: K = \frac{2c}{(1+b^2)^{3/2}} \end{aligned}$$

3.3.3 Casos que producen error

Analicemos ahora lo que ocurriría si la restricción que hemos impuesto de que la parábola no debe cruzar los límites inferior y superior de nuestra ventana 5×3 no se cumpliese. Supongamos por ejemplo que la curva alcance el pixel $(i-1, j+3)$. La situación sería la que se muestra en la figura 3.10.

Esto provocaría que la expresión para el término L (ecuación 3.6), que representaba el área interior a la columna izquierda de la ventana que se encuentra bajo la curva, sea ahora diferente. Más concretamente, el límite izquierdo de la integral sería ahora x_1 en lugar de $-3h/2$ (ver figura 3.10), cuyo valor es

$$x_1 = \frac{-b + \sqrt{b^2 - 4c\left(a + \frac{5}{2}h\right)}}{2c}$$

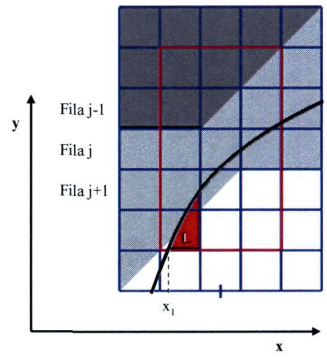


Figura 3.10: La parábola corta el borde inferior de la ventana, en lugar de cortar por el borde izquierdo.

ya que $c < 0$ (puesto que si no la parábola no podría alcanzar el pixel $(i-1, j+3)$ y a la vez pasar con pendiente entre 0 y 1 por el pixel central). Por lo tanto, la nueva expresión para L sería:

$$L = \int_{x_1}^{-h/2} \left(a + bx + cx^2 + 5h/2\right) dx = \left(a + \frac{5}{2}h\right) x + \frac{1}{2}bx^2 + \frac{1}{3}cx^3 \Big]_{x_1}^{-h/2}$$

lo cual resulta en una expresión que ya no es lineal en las variables (a, b, c) como antes. Esto produce que el sistema que ahora obtendríamos tampoco es lineal, por lo que sería bastante más complicado de resolver que el que obtuvimos en la sección anterior. Además, aparte de calcular las expresiones para las incógnitas (a, b, c), también hay que detectar cuándo nos encontramos en este caso, cosa que también sería harto complicada.

En cualquier caso, esta situación sólo sucederá para curvaturas muy grandes (radios muy pequeños, de 4 pixels de radio o menos). Por ejemplo, una circunferencia de radio 4 pixels que cayera en la posición que se ve en la figura 3.11, no podría detectarse con total exactitud.

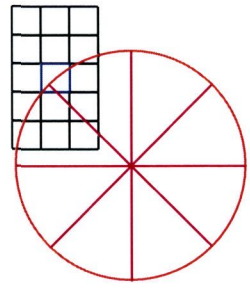


Figura 3.11: Una circunferencia de radio 4 podría cortar el borde inferior de la ventana aún pasando por el pixel central con una pendiente menor que 1 en su vertical central.

Sólo hemos analizado el caso en el que la parábola alcance el pixel $(i-1, j+3)$. Como la parábola dibujada se curva hacia abajo, el borde superior de la ventana nunca podría ser intersectado, puesto que la pendiente en la vertical central es menor que 1. Si la parábola se curvara hacia arriba, entonces sería el borde inferior el que no podría intersectar, y habría que analizar el caso cuando la parábola alcanzara el pixel $(i+1, j-3)$, pero obtendríamos idéntico resultado pues la situación es simétrica.

Radio de curvatura mínimo que garantiza un error nulo Para dar una medida de cuál es el radio de la menor circunferencia posible que nuestro método puede garantizar en cualquier caso, $R_{\min}$, podemos buscar cuál es la mayor circunferencia posible (con pendiente positiva menor que 1 sobre la vertical central) que en el peor caso toque la esquina inferior izquierda. Esta circunferencia será aquélla que toque el punto $P_1 = (0, -h/2)$ con una pendiente de 45°, y pase también por el punto $P_2 = (-3h/2, -5h/2)$, como se ve en la figura 3.12.

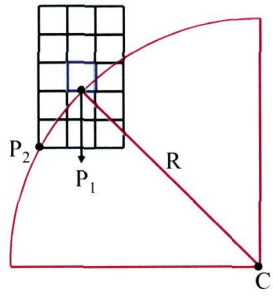


Figura 3.12: La circunferencia mayor en la que el método propuesto puede fallar es aquélla que pasa por el punto P_1 con pendiente 1 y toca el punto P_2.

Para calcular dicho radio, sabemos que el centro de esa circunferencia, $C = (x_c, y_c)$, estará a lo largo de la recta $y = -x - h/2$. Por lo tanto, para encontrar C, buscamos un punto sobre dicha recta que esté a la misma distancia de P_1 y P_2. Estas distancias son:

$$\begin{aligned} \left\|\overrightarrow{P_1C}\right\| &= \sqrt{x_c^2 + \left(y_c + \frac{1}{2}h\right)^2} = \sqrt{2x_c^2} \\ \left\|\overrightarrow{P_2C}\right\| &= \sqrt{\left(x_c + \frac{3}{2}h\right)^2 + \left(y_c + \frac{5}{2}h\right)^2} = \sqrt{2x_c^2 - x_c h + \frac{25}{4}h^2} \end{aligned}$$

Por lo tanto para ecotrar las coordenadas de C resolvemos la ecuación

$$\begin{aligned} 2x_c^2 &= 2x_c^2 - x_c h + \frac{25}{4}h^2 \\ x_c &= \frac{25}{4}h \end{aligned}$$

y finalmente hallamos el radio $R_{\min}$

$$R_{\min} = \sqrt{2x_c^2} = \frac{25}{4}\sqrt{2}h \simeq 8.84h \tag{3.9}$$

Es decir, el método garantiza siempre el resultado correcto para circunferencias cuyo radio sea mayor de $8.84h$ unidades (considerando que h es la longitud de un pixel). Para radios menores existirá la posibilidad de cometer un error en el resultado, pero sólo en aquellas situaciones donde el contorno no cruce la ventana de izquierda a derecha.

Quizás pueda parecer que un radio cercano a 9 pixels sea una cifra demasiado elevada. Sin embargo, téngase en cuenta que aunque para radios de curvatura inferiores a $R_{\min}$ no podamos garantizar un error totalmente nulo en todos los casos, la probabilidad de que una circunferencia de radio inferior pero cercano a $R_{\min}$ corte el borde inferior o superior de la ventana es muy pequeña. De hecho, vamos a calcular a continuación una función de probabilidad que nos indique para cada radio de curvatura en qué casos esto va a ocurrir y en cuáles no, donde se apreciará mejor tal afirmación.

Probabilidad de cometer un error en función del radio de curvatura Consideremos la figura 3.13, donde una circunferencia de radio R, con centro en el punto $C = (x_c, y_c)$, atraviesa el pixel central de la ventana de tal forma que corta a la vertical central de dicho pixel en el punto $P = (0, d)$. Supongamos que d puede tomar cualquier valor del intervalo $[-h/2, h/2]$ con idéntica probabilidad. De igual manera, supongamos que el ángulo α que posee la recta tangente a la circunferencia en P puede tomar cualquier valor del intervalo $[0, \pi/4]$ también con idéntica probabilidad. En este caso, podría calcularse la probabilidad de que la circunferencia no alcance el pixel $(i-1, j+3)$.

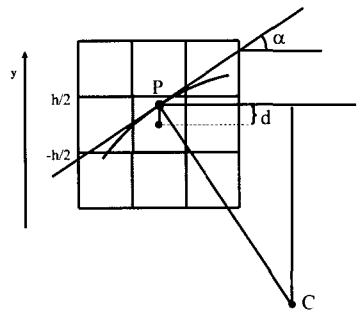


Figura 3.13: La circunferencia con centro C corta la vertical central del pixel central en el punto $P = (0, d)$.

Si dicha probabilidad la calculamos para cualquier valor de R, obtendríamos una función $P(R)$ cuya gráfica puede verse en la figura 3.14 (en el anexo A se muestra el cálculo de la expresión de la función).

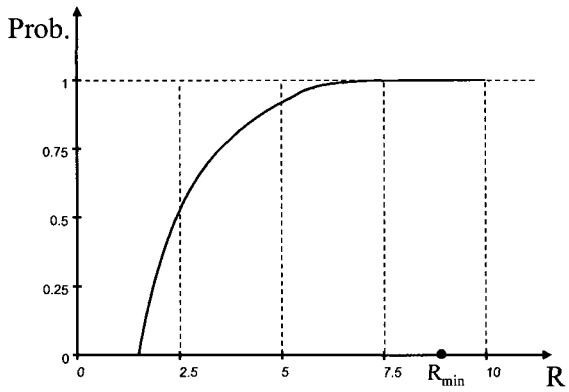


Figura 3.14: Probabilidad de que una circunferencia de radio R no produzca error con nuestro método. El radio está medido en unidades de h.

Como puede apreciarse, para radios inferiores aunque cercanos a $R_{\min}$, la probabilidad de que no haya error es altísima. Por ejemplo, para un radio de curvatura de 5 pixels, la probabilidad de acertar con exactitud es casi el 92%. Para un radio de 4 pixels es casi el 82%. Incluso para un radio de 3 pixels la probabilidad es mayor del 65%. A partir de aquí, ya la probabilidad desciende en picado, llegando a ser 0 total cuando $R = 3h/2$, es decir, cuando es imposible que la circunferencia corte el borde izquierdo de la ventana.

Por otro lado, lo que hemos calculado es solamente la probabilidad de que la circunferencia no toque el borde izquierdo de la ventana, y por lo tanto haya un error en la estimación de los parámetros de la curva. Es decir, cuando dijimos que la probabilidad de acertar con exactitud para un radio de 5 pixels era del 92%, significa que el error será 0 con una probabilidad de 0.92, y diferente de cero con una probabilidad de 0.08. Pero incluso en estos casos, el error no será demasiado alto, ya que es debido a la inexactitud en el valor que toma el área L (ver figura 3.10), el cual será inferior al que realmente debería tener si el borde inferior de la ventana estuviese mucho más abajo. Para radios no demasiado pequeños, el error cometido en la medida de L no es demasiado importante.

Con todo esto queremos indicar que aunque el método garantice la exactitud total solamente para radios de curvatura mayores que 8 pixels, podemos afirmar que para radios un poco menores también se conseguirá detectar los parámetros de la curva, aunque con menor precisión que para radios grandes.

3.3.4 Buscando circunferencias en lugar de parábolas

Podría parecer lógico que, ya que estamos buscando estimar la curvatura del contorno en el pixel, intentásemos encontrar la ecuación de la circunferencia que mejor se aproxima a las intensidades de los pixels dentro de la ventana, en lugar de buscar una parábola. El problema es que buscar una circunferencia da lugar a un sistema de ecuaciones no lineal, complejo de resolver.

Supongamos que en la figura 3.8 la curva a buscar sea una circunferencia, cuya expresión fuese

$$(x - x_c)^2 + (y - y_c)^2 = r^2$$

En este caso, las incógnitas que debemos resolver serían el radio de la circunferencia, r, y las coordenadas de su centro, (x_c, y_c).

Si seguimos el mismo desarrollo usado en la sección 3.3.1 para encontrar parábolas obtendríamos, por ejemplo, para el área bajo la curva interior a la columna izquierda de la ventana, L, la siguiente expresión

$$\begin{aligned} L &= \int_{-3h/2}^{-h/2} \left(y_c + \sqrt{r^2 - (x - x_c)^2} + 5h/2 \right) dx = \\ &= \left[\frac{r^2}{2} \arcsin\left(\frac{x - x_c}{r} \right) + \frac{x - x_c}{2} \sqrt{r^2 - (x - x_c)^2} + y_c x \right]_{-3h/2}^{-h/2} \end{aligned}$$

Para las otras áreas, M y R, obtendríamos expresiones similares, todas ellas lejos de ser lineales en las variables (x_c, y_c, r). Podría tratar de resolverse mediante algoritmos numéricos, pero entonces ya estaríamos hablando del método expuesto en la sección 3.1.3.

Error máximo cometido En cualquier caso, el error cometido por aproximar el contorno utilizando una parábola en lugar de una circunferencia es muy pequeño. Profundizaremos bastante más en esta cuestión en capítulos posteriores, e incluso llegaremos a corregir este error. Pero por ahora nos bastará con calcular el error cometido en el peor caso posible, que es el comentado anteriormente en la sección anterior (ver figura 3.12), para hacernos una idea de hasta dónde puede llegar la magnitud del error. Sabemos que éste es el peor caso porque la mayor diferencia entre ambos tramos de curva se produce para radios pequeños y pendientes alejadas de la horizontal.

Sea por tanto una circunferencia de radio $r = R_{\min} = 25\sqrt{2}/4h$, que pase por el punto $P_1 = (0, -h/2)$ con pendiente 1 y toca el punto $P_2 = (-3h/2, -5h/2)$. Y empleemos nuestro método para ver el error cometido. Tomemos los valores $h = 1$, $A = 250$, $B = 150$ para obtener resultados numéricos.

En primer lugar, las áreas bajo la circunferencia interiores a cada columna tendrían realmente las siguientes expresiones:

$$\begin{aligned} L &= \int_{-3h/2}^{-h/2} \left(y_c + \sqrt{r^2 - (x - x_c)^2} + 5h/2 \right) dx \simeq 0.780\,67 \\ M &= \int_{-h/2}^{h/2} \left(y_c + \sqrt{r^2 - (x - x_c)^2} + 5h/2 \right) dx \simeq 1.986\,7 \\ R &= \int_{h/2}^{3h/2} \left(y_c + \sqrt{r^2 - (x - x_c)^2} + 5h/2 \right) dx \simeq 2.851\,9 \end{aligned}$$

Usando ahora el lema 3.2, obtendríamos los valores de A y B correctos, ya que los pixels utilizados para evaluar su valor no se ven influidos. Para los coeficientes de la parábola que mejor se aproxima a los valores de intensidad que tenemos, obtendríamos los siguientes valores numéricos

$$\begin{aligned} a &\simeq -0.499\,25 \\ b &\simeq 1.035\,6 \\ c &\simeq -0.170\,58 \end{aligned}$$

lo cual nos daría los siguientes resultados

$$\begin{aligned} \text{corte con la vertical central del pixel} &\simeq (0, -0.499\,25) \\ \text{pendiente en } P_1 &: b \simeq 1.035\,6 \\ \text{radio de curvatura} &: \left|\frac{1}{K}\right| \simeq 8.750\,3 \end{aligned}$$

El error del desplazamiento sería inferior al 0.1% del tamaño del pixel. El error relativo de la curvatura sería menor al 1%. El error mayor se produciría en la pendiente, habiendo una diferencia entre ambas pendientes de 1.00 grados. De hecho, si dibujásemos ampliado este caso (ver figura 3.15), apenas se notaría la diferencia entre ambas curvas en el interior de la ventana.

3.3.5 Comparativa con el método de primer orden

Observando las expresiones obtenidas para los coeficientes de la recta y de la parábola en los métodos de primer y segundo orden respectivamente (lemas 3.1 y 3.2), puede apreciarse que el valor de b es idéntico en ambos. Esto significa que no existe diferencia en la estimación de la orientación del contorno entre uno y otro método.

Por otro lado, el valor de a en el de segundo orden puede expresarse como

$$a = \frac{2S_M - 5\,(A + B)}{2(A - B)} - \frac{1}{12}c$$

Es decir, la misma expresión que para el caso lineal, a la que se le resta un doceavo del valor de c. Esto significa que en el caso de contornos rectos, c valdrá cero, y ambos desplazamientos coincidirán. Sin embargo, cuando el contorno tenga una cierta curvatura, c será distinto de cero, con lo cual el desplazamiento estimado por el método lineal tendrá un error proporcional al valor de c.

Puede parecer obvio que al ser el método de segundo orden más general que el de primer orden, debería usarse en todos los casos. Sin embargo, cuando el objetivo de un cierto estudio sea localizar líneas rectas, usar el método líneal dará valores más precisos, ya que éste tratará siempre de encontrar la mejor recta posible, mientra que el método cuadrático puede encontrar contornos de cierta curvatura debido al ruido procedente de la adquisición de la imagen.

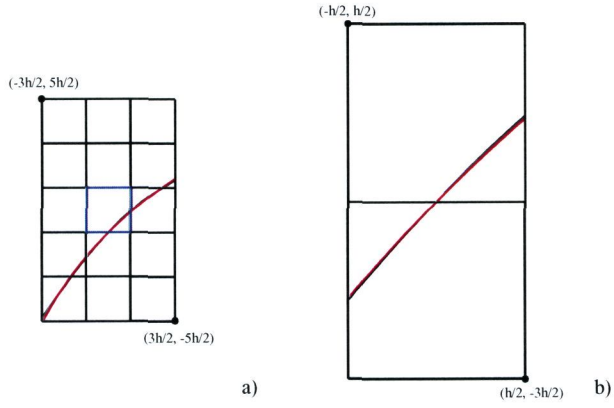


Figura 3.15: a) En color rojo aparece una circunferencia de radio $R_{\min}$ que pasa por los puntos $(0, -h/2)$ y $(-3h/2, -5h/2)$. En negro aparece la parábola estimada por nuestro método para el contorno que atraviesa el pixel central. b) Ampliación de la columna central de la ventana

3.4 Generalización para el resto de octantes

Al igual que en el capítulo anterior, antes de desarrollar el algoritmo hay que generalizar la situación para los 8 octantes, ya que sólo hemos tenido en cuenta los casos en donde la pendiente estaba entre 0 y 1 y $A > B$, siendo A la intensidad por debajo de la curva, y B por encima (ver caso 1 en la figura 3.16a).

Cada caso se divide además en dos subcasos, atendiendo a la concavidad o convexidad de la curva, es decir, al signo del coeficiente c. En la figura 3.16(b y c) se ven los dos subcasos para el octante 1. En el primero, $c < 0$ y por tanto $K < 0$, mientras que en el segundo, $c, K > 0$. Nótese que los signos para la curvatura K y para el coeficiente del término cuadrático, c, siempre coinciden, debido a la ecuación 3.8. Esto significa que el signo de la curvatura da información de hacia qué lado se dirige la curva, siendo negativa cuando la curva se estira hacia el lado más claro (figura 3.16b), y positiva cuando lo hace hacia el lado más oscuro (figura 3.16c).

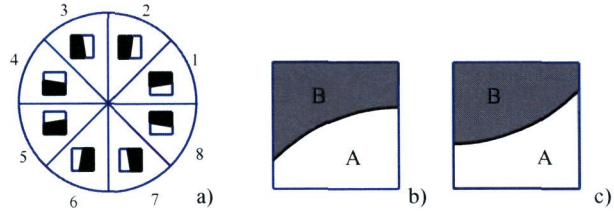


Figura 3.16: a) El contorno de una imagen pertenecerá siempre a uno de estos 8 octantes. b) Octante 1 para $c, K < 0$. c) Octante 1 para $c, K > 0$.

Por otro lado, en ambos subcasos, el vector normal apuntaría hacia abajo y a la derecha, indicando que en esa dirección es hacia donde crece la intensidad en la imagen (ya que $b > 0$ en la ecuación 3.7).

Analicemos ahora el octante 8, donde la pendiente está entre 0 y -1. Vemos que las expresiones para las

sumas de cada columna (ecuaciones 3.3) se mantienen válidas, pero no así las expresiones para estimar los valores de las intensidades A y B (ecuaciones 3.4). Dichas expresiones se basaban en utilizar aquellos pixels por donde sabíamos que el contorno no cruzaría, y es por eso que para el octante 1 utilizabamos los tres pixels de la esquina inferior derecha para A y los tres de la esquina superior izquierda para B. Sin embargo, en este octante 8, sería mejor utilizar la esquina inferior izquierda para A y la superior derecha para B, quedando las nuevas expresiones como sigue:

$$\begin{aligned} A &= \frac{F_{i,j+2} + F_{i-1,j+2} + F_{i-1,j+1}}{3} \\ B &= \frac{F_{i+1,j-1} + F_{i+1,j-2} + F_{i,j-2}}{3} \end{aligned}$$

Las expresiones para los coeficientes de la parábola, (a, b, c), quedarían igual, cumpliéndose en este caso que $b < 0$. El resto de las expresiones (desplazamiento d, vector normal N y curvatura K) quedarían exactamente igual. Nótese que ahora el vector normal apuntaría hacia abajo a la izquierda.

Analicemos ahora los octantes 4 y 5. Podría parecer que todas las expresiones serían las mismas, con lo cual A y B seguirían siendo las intensidades por debajo y encima de la curva respectivamente, sólo que ahora $A < B$. Sin embargo, hay que añadir un pequeño cambio.

Supongamos que tenemos la imagen de un círculo claro sobre fondo oscuro, como el que se ve en la figura 3.17. En el recuadro azul 1, perteneciente al octante 1, A sería mayor que B, el vector normal apuntaría hacia abajo, y la curvatura sería negativa. Lo coherente sería que en el recuadro 5, perteneciente al octante 5, el vector normal apunte hacia arriba y que la curvatura siga siendo negativa, ya que la curva sigue el mismo patrón en ambos recuadros. Sin embargo, la curva nos saldría con un valor de c positivo, ya que la gráfica de la curva dentro del pixel es idéntica a la de la figura 3.16c) para el primer octante. Esto significa que cuando $A < B$, el signo de K debe ser invertido.

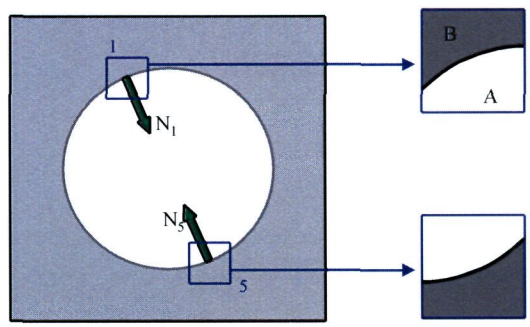


Figura 3.17: En cada pixel borde, el vector normal debe apuntar hacia el centro de la circunferencia, y la curvatura debe ser negativa.

Con esto ya podemos llegar al siguiente lema para todos los casos en los que $b \in [-1, 1]$:

Lema 3.3 *Sea $F(i, j)$ una imagen por cuyo pixel (i, j) pasa un contorno que separa dos zonas de intensidad A (debajo del contorno) y B (encima del contorno), y donde se cumple que*

$$|F_y(i, j)| > |F_x(i, j)|$$

Si suponemos que en la vecindad de ese pixel, el borde se comporta como una curva de segundo grado, podemos estimar los parámetros de dicho borde (posición, pendiente, magnitud del salto de intensidad y curvatura,

medidos sobre la vertical central del pixel), calculando previamente la parábola que mejor se aproxima a dicho borde. Los pasos son los siguientes:

a) obtenemos las sumas de las columnas izquierda, central y derecha de una ventana 5×3 *centrada en el pixel* (i, j)*, cuyas expresiones respectivas son:*

$$\begin{aligned} S_L &= \sum_{n=j-2}^{j+2} F_{i-1,n} \\ S_M &= \sum_{n=j-2}^{j+2} F_{i,n} \\ S_R &= \sum_{n=j-2}^{j+2} F_{i+1,n} \end{aligned}$$

b) obtenemos también una estimación para los valores de intensidad a ambos lados del borde:

$$\begin{aligned} A &= \frac{F_{i,j+2} + F_{i+m,j+2} + F_{i+m,j+1}}{3} \\ B &= \frac{F_{i-m,j-1} + F_{i-m,j-2} + F_{i,j-2}}{3} \end{aligned}$$

donde

$$m = \begin{cases} 1 & \text{si } F_x(i,j)F_y(i,j) > 0 \\ -1 & \text{si } F_x(i,j)F_y(i,j) < 0 \end{cases}$$

c) a continuación detectamos los coeficientes de la parábola $y = a + bx + cx^2$ *como sigue:*

$$\begin{aligned} a &= \frac{26S_M - S_L - S_R - 60(A+B)}{24(A-B)} \\ b &= \frac{S_R - S_L}{2(A-B)} \\ c &= \frac{S_L + S_R - 2S_M}{2(A-B)} \end{aligned}$$

d) finalmente calculamos los parámetros de la parábola

$$\begin{aligned} \text{corte con la vertical central del pixel} &: (0, a) \\ \text{vector normal} &: N = \frac{A-B}{\sqrt{1+b^2}}[b, -1] \\ \text{curvatura} &: K = \frac{2cn}{(1+b^2)^{3/2}} \end{aligned}$$

donde

$$n = \begin{cases} 1 & \text{si } F_y(i,j) > 0 \\ -1 & \text{si } F_y(i,j) < 0 \end{cases}$$

Para los otros cuatro octantes, en donde la pendiente es mayor que 1 en valor absoluto, se establece un esquema idéntico donde las variables x e y se ven intercambiadas, y se trabaja con las filas de la imagen en lugar de con las columnas. Ahora el objetivo será encontrar la curva $x = a + by + cy^2$ que representa el contorno. A continuación mostramos el lema para estos 4 octantes:

Lema 3.4 *Sea $F(i,j)$ una imagen por cuyo pixel (i,j) pasa un contorno que separa dos zonas de intensidad A (debajo del contorno) y B (encima del contorno), y donde se cumple que*

$$|F_y(i,j)| < |F_x(i,j)|$$

Si suponemos que en la vecindad de ese pixel, el borde se comporta como una curva de segundo grado, podemos estimar los parámetros de dicho borde (posición, pendiente, magnitud del salto de intensidad y curvatura, medidos sobre la horizontal central del pixel), calculando previamente la parábola que mejor se aproxima a dicho borde. Los pasos son los siguientes:

a) obtenemos las sumas de las filas superior, central e inferior de una ventana 3×5 centrada en el pixel (i,j), cuyas expresiones respectivas son:

$$\begin{aligned} S_L &= \sum_{n=i-2}^{i+2} F_{n,j-1} \\ S_M &= \sum_{n=i-2}^{i+2} F_{n,j} \\ S_R &= \sum_{n=i-2}^{i+2} F_{n,j+1} \end{aligned}$$

b) obtenemos también una estimación para los valores de intensidad a ambos lados del borde:

$$\begin{aligned} A &= \frac{F_{i+2,j} + F_{i+2,j+m} + F_{i+1,j+m}}{3} \\ B &= \frac{F_{i-1,j-m} + F_{i-2,j-m} + F_{i-2,j}}{3} \end{aligned}$$

donde

$$m = \begin{cases} 1 & \text{si } F_x(i,j)F_y(i,j) > 0 \\ -1 & \text{si } F_x(i,j)F_y(i,j) < 0 \end{cases}$$

c) a continuación detectamos los coeficientes de la parábola $x = a + by + cy^2$ como sigue:

$$\begin{aligned} a &= \frac{26S_M - S_L - S_R - 60(A+B)}{24(A-B)} \\ b &= \frac{S_R - S_L}{2(A-B)} \\ c &= \frac{S_L + S_R - 2S_M}{2(A-B)} \end{aligned}$$

d) finalmente calculamos los parámetros de la parábola

$$\begin{aligned} \text{corte con la horizontal central del pixel} &: (a, 0) \\ \text{vector normal} &: N = \frac{A-B}{\sqrt{1+b^2}}[1, -b] \\ \text{curvatura} &: K = \frac{2cn}{(1+b^2)^{3/2}} \end{aligned}$$

donde

$$n = \begin{cases} 1 & \text{si } F_x(i,j) > 0 \\ -1 & \text{si } F_x(i,j) < 0 \end{cases}$$

3.5 Algoritmo para la localización de contornos de segundo orden

Con los lemas anteriores ya estamos en disposición de desarrollar el algoritmo para tratar de detectar todos los parámetros del borde (posición, pendiente, salto de intensidad y curvatura) en una imagen. Al igual que en el método lineal, lo primero es detectar en la imagen qué pixels podemos etiquetar como borde, a partir de las imágenes de parciales F_x y F_y, generadas mediante una convolución con sendas máscaras H_x y H_y. Estamos interesados en aquellos pixels que cumplan

$$F_x^2(i,j) + F_y^2(i,j) > \delta^2$$

donde δ es un umbral que habrá que prefijar, y que además cumplan que

$$|F_y(i,j-2)| < |F_y(i,j-1)| < |F_y(i,j)| > |F_y(i,j+1)| > |F_y(i,j+2)|$$

en el caso en que $|F_y(i,j)| > |F_x(i,j)|$, o bien que

$$|F_x(i-2,j)| < |F_x(i-1,j)| < |F_x(i,j)| > |F_x(i+1,j)| > |F_x(i+2,j)|$$

en el caso en que $|F_y(i,j)| < |F_x(i,j)|$. Esta última condición era para evitar puntos aislados que no correspondan a un perfil como el que se estudió en la sección 2.1.1.

Una vez tenemos los pixels identificados, aplicaremos alrededor de cada uno el esquema desarrollado en la sección anterior para obtener los parámetros estimados para el contorno. Esto significa que, para cada pixel borde, el método usará los valores de un entorno centrado en dicho pixel, cuyas dimensiones serán 5×3 (para el caso en el que $|F_y(i,j)| > |F_x(i,j)|$) ó 3×5 en caso contrario.

De nuevo deben cumplirse tres condiciones dentro de dicho entorno, o al menos acercarse lo más posible, para que la situación alrededor de esos pixels sea lo más parecida a la situación ideal sobre la que se ha desarrollado el método. Dichas condiciones son:

1) Dentro de ese entorno, el contorno puede ser aproximado como una curva de grado 2.

2) Dentro de ese entorno, no debe aparecer ningún otro contorno diferente.

3) Dentro de ese entorno, la intensidad a ambos lados del contorno debe ser lo más homogénea posible.

Habría una cuarta condición más, que sería que la curvatura no fuese excesivamente grande, debido al error que podría producirse, tal y como se vio en la sección 3.3.3. Como ya se comentó para el método lineal, es difícil que las cuatro condiciones se cumplan exhaustivamente en todos los pixels borde, pero también es bastante probable que se cumplan en un grado alto. La primera condición fallará en zonas donde haya esquinas. La segunda fallará en zonas donde haya dos bordes muy cercanos entre sí. La tercera fallará en los bordes de zonas con textura muy fina. En los demás casos, las condiciones se cumplirán.

Hay otra condición que también asumimos, y es que la imagen no tenga ruido, o que tenga muy poco. En el próximo capítulo abordaremos este problema, pero por ahora trabajaremos con imágenes tan limpias como sea posible.

El pseudo-código de nuestro algoritmo podría ser el siguiente:

Funcion Contornos2 (delta)

```
Calcular Fx, Fy
Para todos los pixels (i,j) de la imagen
   Si Fx(i,j)*Fx(i,j) + Fy(i,j)*Fy(i,j) < delta * delta Continuar
   Si |Fy(i,j)| > |Fx(i,j)| Entonces
      Si |Fy(i,j)| no es maximo en su columna Continuar
      Calcular Curvatura, Vector Normal y Posicion segun Lema 3.3
         Si No
      Si |Fx(i,j)| no es maximo en su fila Continuar
      Calcular Curvatura, Vector Normal y Posicion segun Lema 3.4
   Fin Si
Fin Para
```

3.6 Ejemplos

A continuación vamos a estudiar el comportamiento de nuestro método en imágenes sintéticas y reales, haciendo hincapié en la estimación de la curvatura.

3.6.1 Contorno ideal

Comencemos por la imagen 2.22 generada sintéticamente según nuestras hipótesis de trabajo iniciales. En dicha imagen aparecía un contorno recto ideal que cruzaba toda la imagen. Ya vimos en el capítulo previo que usando el método lineal propuesto, tanto el desplazamiento sub-pixel del contorno, como la pendiente, y la magnitud del salto de intensidad a ambos lados, estimados para todos los pixels borde, era exactamente el usado para la generación de la imagen, sin ningún tipo de error. Si aplicásemos el método cuadrático, los valores seguirían siendo exactos, y además nos saldría una curvatura prácticamente cero (del orden de 10^{-10}) para todos los pixels borde.

Apliquemos ahora nuestro método a un círculo también generado sintéticamente, como el que aparecía en la figura 2.23, de radio 20 pixels e intensidad 255 en el interior y 0 en el exterior. En la tabla 3.1 se indica el error numérico cometido al aplicar nuestro método cuadrático a dicha imagen, junto con los resultados de los métodos anteriores. El error del módulo viene medido en unidades de intensidad, el de la dirección en grados, y el de la posición en porcentaje relativo al tamaño del pixel. Es decir, con el método cuadrático no se aprecia error alguno para el salto de intensidad, el error medio para la orientación del contorno es de 0.05 grados, y el error medio cometido en la posición del contorno es un 0.001% de la longitud del pixel.

La bajada del error en cuanto a la posición del contorno, con respecto al método lineal, es consecuencia de trabajar con una mejor aproximación del contorno (curva de grado dos en lugar de una recta). El problema del método lineal es que a mayor curvatura del contorno real que se encuentre en el pixel, más desplazada será la recta que se obtiene con respecto a su verdadera posición. Lo podemos ver mejor con el siguiente ejemplo, que se muestra en la figura 3.18.

Supongamos que queremos aplicar el método lineal para detectar los parámetros del contorno que cruza el pixel central (i, j) (dibujado en azul), y para ello usaremos una ventana 5×3 centrada en dicho pixel, marcada en verde. Sea una circunferencia de radio 5 centrada en el centro del pixel $(i + 3, j + 4)$. Dicha circunferencia

	Err. módulo		Err. dirección		Err. posición		Radio de curvatura		
Método	Media	Max.	Media	Max.	Media	Max.	Media	Min.	Max.
Tradic. $\alpha = 0.5$	6.34	17.2	0.80	1.70	*NC*	*NC*	*NC*	*NC*	*NC*
MPG1	0.00	0.00	0.05	0.17	0.29	0.57	*NC*	*NC*	*NC*
Segundas deriv.	3.22	4.31	0.81	1.72	*NC*	*NC*	28.32	12.49	32.45
Expresión analítica	2.12	3.32	0.45	1.27	*NC*	*NC*	24.32	15.69	25.43
MPG2	0.0	0.0	0.05	0.17	0.001	0.006	19.98	19.96	19.98

Tabla 3.1: Errores cometidos por cada método al obtener el contorno de una circunferencia de radio 20. MPG1, MPG2: métodos propuestos de grado 1 y 2. NC: no lo calcula

pasará exactamente por el centro del pixel $(0, 0)$. El valor estimado para el coeficiente a por nuestro método es aproximadamente -0.0164, con lo cual la recta obtenida se encuentra ligeramente desplazada hacia abajo (un 1.6% de la longitud del pixel), como se puede apreciar en el dibujo ampliado del pixel.

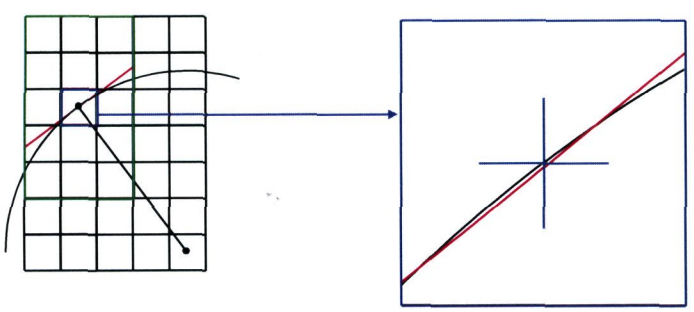


Figura 3.18: Aplicación del método lineal sobre el pixel $(0, 0)$ (en azul) para obtener la recta (en rojo) que mejor se aproxima a un contorno formado por una circunferencia de radio 5 centrada en el centro del pixel $(3, 4)$ (en negro). A la derecha se muestra una ampliación del pixel $(0, 0)$.

En cuanto a los resultados de la curvatura en la tabla 3.1, se ha creído más conveniente mostrar su inversa, es decir, el radio de curvatura, e indicar cuál fue el valor medio para todos los pixels borde, así como los valores mínimo y máximo que se obtuvieron. Como puede apreciarse, el valor medio es de 19.98 unidades de distancia para el radio, siendo la peor estimación 19.96 unidades. Esto significa un error relativo del 0.1% en la estimación del radio.

Hay que tener en cuenta la importancia de este resultado, ya que el valor de la curvatura es altamente sensible, y para cada pixel estamos usando exclusivamente una vecindad de $5 \times 3 = 15$ pixels vecinos para estimar la curvatura del contorno. En la figura 3.19 pueden verse varios tramos de circunferencia con distintos radios, y puede apreciarse la poca diferencia entre ellos a pesar de que sus radios son bastante diferentes. Nótese como variaciones muy pequeñas en las intensidades de los pixels vecinos pueden provocar grandes variaciones en el radio de curvatura estimado.

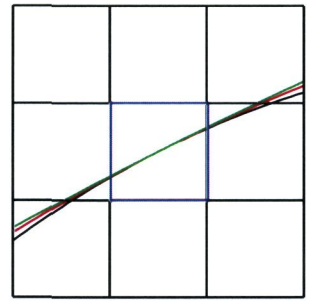


Figura 3.19: Arcos de circunferencia con distinto radio: 10 (negro), 20 (rojo) y 50 (verde) pixels de radio, pintados sobre una ventana de 3×3 pixels.

3.6.2 Contorno circular semi-ideal

Usemos ahora la imagen semi-ideal de la diana, usada en el capítulo anterior (ver figura 2.25). Como estamos interesados en obtener valores de curvatura, esta vez nos centraremos en los pixels que forman el borde de las dos circunferencias concéntricas que aparecen en la imagen, tal y como se muestra en la figura 3.20. Ya que una de las condiciones de nuestro método es que en la ventana centrada en el pixel donde se toman los valores, no debe aparecer más de un contorno, descartaremos en el cálculo los pixels de la circunferencia cercanos a las líneas rectas que dividen los sectores, ya que en esos puntos los valores no serán correctos.

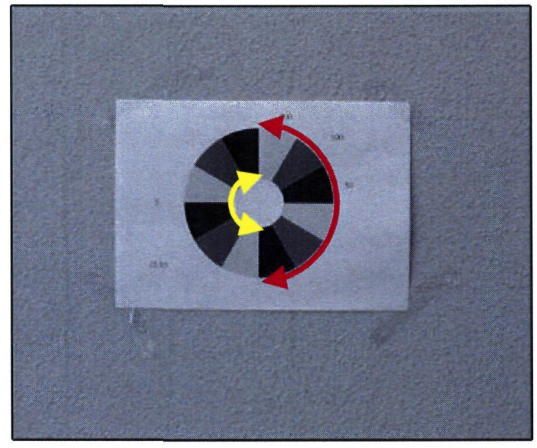


Figura 3.20: Se van a estimar los parámetros del contorno con nuestro método cuadrático en los pixels de las circunferencias exterior (en rojo) e interior (en amarillo).

Los resultados aparecen en la tabla 3.2. Puede observarse que los valores estimados para el radio de curvatura son bastante precisos en ambas circunferencias. Nótese también como el signo de la curvatura también ha sido el correcto en ambos casos. Para la circunferencia exterior, la intensidad dentro del círculo es menor que la de

	Num. Pixels	Curv. media	Radio Curv.	Radio aprox.^{2}
Circ. exterior	154	0.0191	52.36	53
Circ. interior	36	−0.0583	−17.15	−18

Tabla 3.2: Curvatura media en las dos circunferencias de la imagen de la diana.

fuera, ya que ésta siempre es blanca, y la de los sectores es más oscura. Por lo tanto, según se comentó con anterioridad, una curvatura positiva implica que la curva tiende a estirarse hacia el color más oscuro. Para la circunferencia interior ocurre justamente lo contrario, ya que el color más claro es ahora el de dentro de la circunferencia. Por tanto, la curva se estira hacia el color más claro, y la curvatura ha de ser negativa.

3.6.3 Imagen real digitalizada

Como último ejemlo, usemos la imagen del popeye del capítulo anterior, y centrémonos en los dos botones inferiores de su camisa (ver figura 3.21). Ambos tienen forma circular, de unos 8 pixels de diámetro aproximadamente cada uno (radio 4). Si aplicamos nuestro método obtenemos las curvaturas medias -0.2644 para el botón de arriba y -0.2773 para el de abajo, lo que significa que los radios de curvatura medio estimados para ambos son -3.78 y -3.60 (el de abajo ligeramente menor). Vemos que el resultado es bastante acertado. También el signo es correcto, siendo negativo, ya que la curva se estira hacia el lado más claro.

También podemos calcular la desviación estándar, para ver entre qué valores de curvatura se mueve la mayoría de los pixels. Para el botón de arriba obtenemos una desviación de 0.142, y para el de abajo 0.127. Esto significa que, la mayor parte de los pixels borde del botón de arriba, obtuvo un valor de curvatura dentro del intervalo $[-0.406, -0.122]$, o lo que es lo mismo, un radio de curvatura entre -8.20 y -2.46. Aunque pueda parecer una oscilación importante, el dibujo de ambas curvas en el interior del pixel es bastante parecido.

3.7 Conclusiones

Se ha desarrollado un esquema capaz de encontrar en imágenes ideales sintéticas la orientación, curvatura y localización sub-pixel de un contorno de segundo grado de forma exacta, así como el salto de intensidad producido a ambos lados del borde. Posteriormente, y a partir de dicho esquema, hemos desarrollado un método para localizar los contornos sobre una imagen real, asumiendo que localmente dichos contornos se comportan como curvas de segundo grado, y hemos visto varios ejemplos prácticos de aplicación.

Hasta ahora todas las imágenes que hemos usado como ejemplos eran bastante limpias, con poco o ningún ruido. Pero normalmente en casos reales la imagen puede no venir con todos sus valores exactos. En el próximo capítulo desarrollaremos un nuevo esquema que nos permitirá lograr mejores estimaciones que el método actual cuando en la imagen exista algo de ruido.

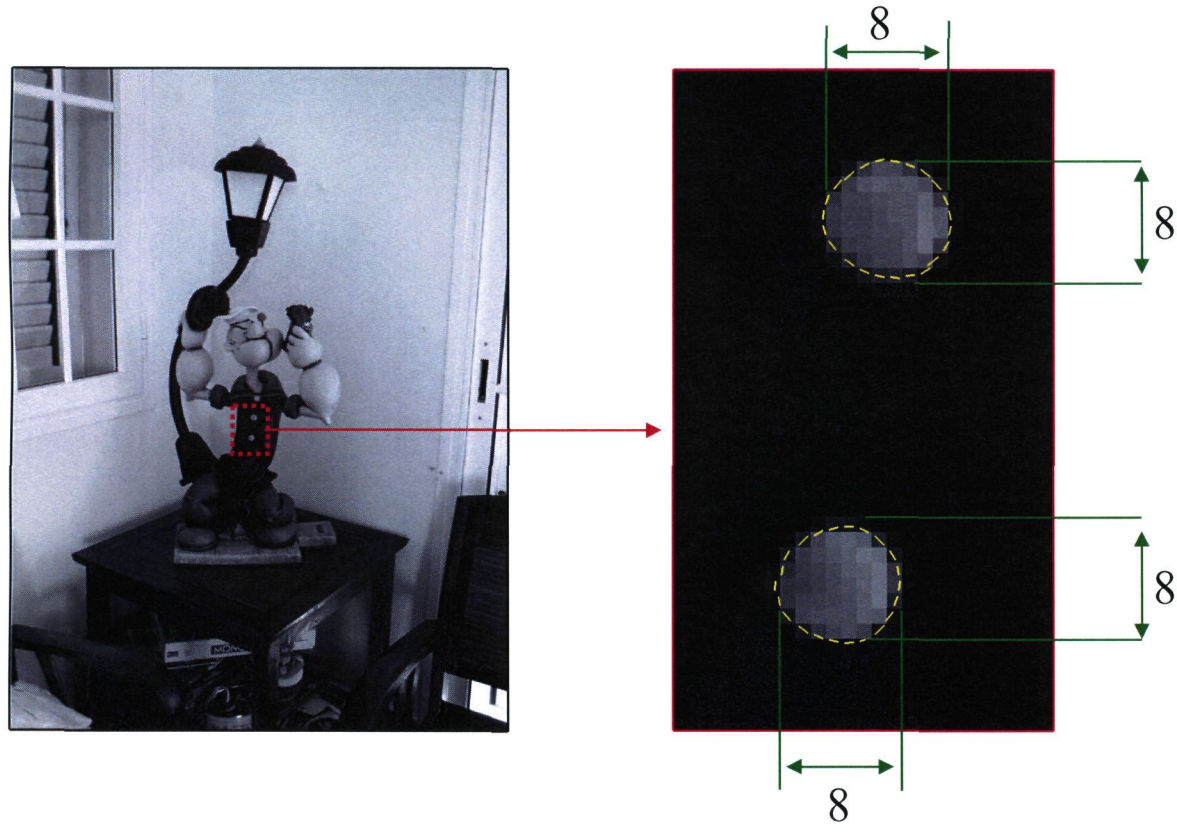


Figura 3.21: Cálculo del contorno sobre la zona de los botones del popeye.

Capítulo 4

Contornos en imágenes suavizadas

- En imágenes reales, el esquema no es tan preciso.
- Por eso los demás métodos suavizan antes la imagen. ¿Por qué nosotros no?
- Bueno, habría que cambiar la hipótesis de partida, para tomar como entrada la imagen suavizada.....

OCTUBRE 2001

Los métodos que hemos desarrollado hasta ahora (de primer y segundo orden) funcionan a la perfección para imágenes ideales, pero producen pequeños errores cuando son aplicados en imágenes reales. Esto es debido al ruido que siempre existe en este tipo de imágenes. La forma clásica de eliminar ruido es convolucionar con un filtro paso bajo, a costa de perder definición en los bordes, que se vuelven más difuminados [SON98]. Sin embargo, si aplicamos nuestras técnicas a una imagen previamente suavizada, no obtendremos los valores exactos, pues la hipótesis de partida de nuestro trabajo (la intensidad de los pixels pertenecientes a los bordes de un objeto es proporcional al área de pixel cubierto por dicho objeto) ya no se cumple en la nueva imagen.

Esto puede verse en la figura 4.1, donde F es una imagen ideal con un contorno de primer orden, tal y como se definió en el capítulo 2 (ecuación 2.1), F' es una versión con ruido de dicha imagen, y G y G' son las imágenes resultantes de suavizar F y F' respectivamente. Nuesto método funciona a la perfección al aplicarlo a la imagen F, detectando con total exactitud los parámetros del contorno. Sin embargo, al aplicarlo a la imagen F' obtenemos un error, debido al ruido existente. Una solución sería suavizar F' para obtener G', la cual tiene menos ruido, y aplicar nuestro método en G'. Pero también habría un error pues, incluso aplicando nuestro método en G, no obtendríamos de forma exacta los parámetros del contorno, ya que la imagen no cumple las hipótesis de partida.

Por tanto, el objetivo del presente capítulo consiste en estudiar cómo es el comportamiento de una imagen con borde previamente suavizada, y así poder desarrollar un método que, aplicado a la imagen suavizada G, detecte de forma exacta el borde original presente en F. La gran ventaja de trabajar con una imagen suavizada es que **el parecido entre el suavizado de una imagen con ruido, G', y el suavizado de la misma imagen sin ruido, G, es mucho mayor que el que existe entre ambas imágenes antes de suavizar**, F y F', con lo cual las estimaciones obtenidas para los parámetros del borde serán más precisas.

Por otro lado, a efectos de comparación con otras técnicas detectoras de borde, hay que considerar que la mayor parte de ellas suavizan la imagen antes de realizar ningún cálculo, para que los resultados salgan más homogéneos. Esto es debido a que, en presencia de ruido, el cálculo de parámetros locales tales como la normal al contorno o la curvatura puede producir resultados bastante alterados. El proceso de suavizado permite

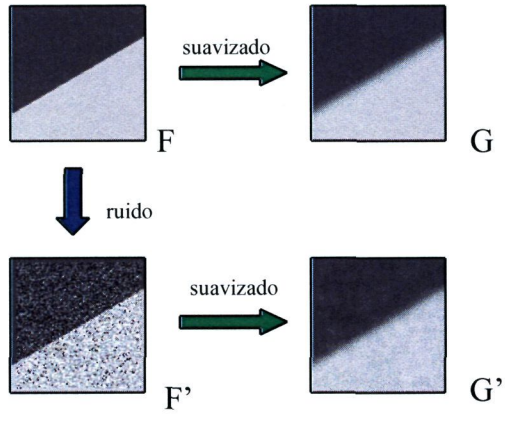


Figura 4.1: F: imagen original. F': imagen con ruido. G y G': suavizados de F y F' respectivamente.

entonces que los valores de la nueva imagen sean más consistentes. Sin embargo, así como este proceso supone una buena ayuda para estas técnicas, para la nuestra ya hemos visto que supondría todo lo contrario. Por lo tanto, y para poder llevar a cabo una comparación de nuestro método con otros, es preciso adecuarlo primero para poder aplicarlo a una imagen suavizada.

La estructura de este capítulo es la siguiente: primero analizaremos las técnicas tradicionales de suavizado de imágenes, para posteriormente desarrollar un esquema que nos permita calcular la localización, orientación y curvatura de los contornos de una imagen F, tomando como dato de entrada la imagen suavizada G. A continuación se verán algunos ejemplos de aplicación sobre imágenes sintéticas y reales, y también sobre imágenes cuyos contornos aparezcan desenfocados.

4.1 Suavizado tradicional de imágenes

Todo proceso de captura y digitalización de una señal puede producir errores en los valores adquiridos. Este fenómeno es lo que se denomina *ruido* en la señal. En la figura 4.2a y 4.2b puede verse una señal unidimensional ideal y otra con ruido respectivamente.

Si se transforma la señal, $f(x)$, al dominio de la frecuencia (mediante una transformada de Fourier), $F(w)$, se puede apreciar cómo el ruido pertenece a la zona de altas frecuencias. Por lo tanto, la técnica clásica para tratar de disminuir sus efectos es aplicar a la imagen un filtro paso bajo [JAI89]. En la figura 4.3a) puede verse el perfil de un filtro paso bajo ideal, $H(w)$, que permite el paso de las frecuencias con valores cercanos a cero, y anula las frecuencias mayores.

De acuerdo con la teoría del filtrado de señales, este filtrado se puede realizar en el dominio de frecuencias mediante la multiplicación de la transformada de la señal, $F(w)$, con el filtro $H(w)$. Este proceso es equivalente a realizar la convolución de la señal original, $f(x)$, en el dominio espacial con la antitransformada del filtro, $h(x)$ (figura 4.3b). Es decir,

$$G(w) = F(w)H(w) \Longleftrightarrow g(x) = f(x) * h(x)$$

donde $G(w)$, $F(w)$, $H(w)$ son las transformadas de Fourier de las señales unidimensionales $g(x)$, $f(x)$ y $h(x)$ respectivamente. Si aplicásemos un filtro de este tipo, obtendríamos una señal más suavizada, como la que se puede ver en la figura 4.2 (d).

Aunque en algunas ocasiones se suele trabajar directamente en el dominio de frecuencias, suele ser más rápido realizar el filtrado en el dominio espacial, convolucionando la señal $f(x)$ con la antitransformada del

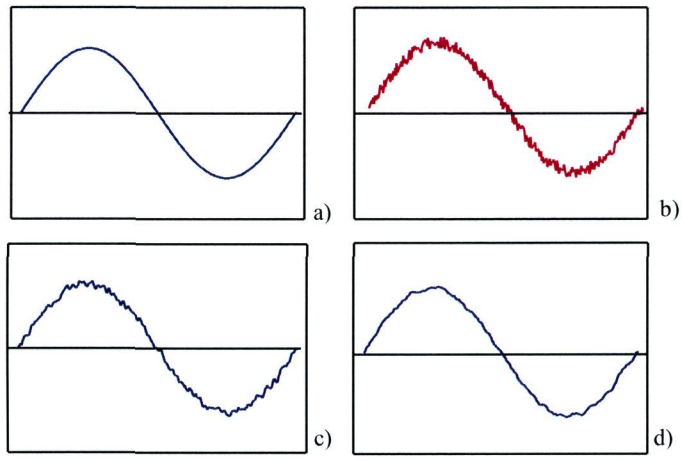


Figura 4.2: a) Señal sin ruido. b) Señal con ruido. c,d) Señal suavizada con distintos filtros

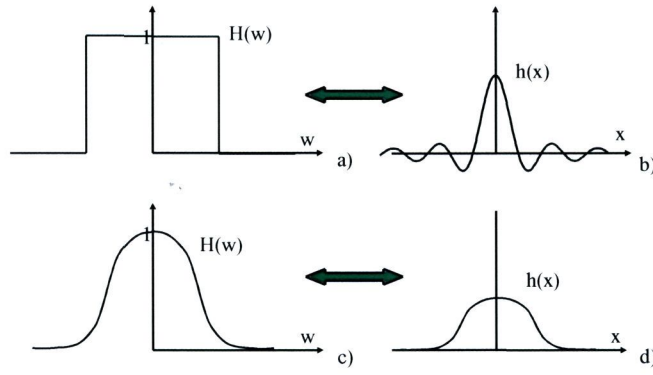


Figura 4.3: Filtro paso bajo ideal: a) dominio frecuencial. b) dominio espacial. Filtro paso bajo no ideal: c) dominio frecuencial. d) dominio espacial.

filtro, $h(x)$. El coste computacional de la convolución es muy inferior a realizar la transformada rápida de Fourier (FFT), siempre y cuando los valores no nulos de $h(x)$ no se alejen en exceso del eje $x = 0$. Es decir, cuanto más ancha sea la zona de valores significativos de $h(x)$, mayor será el coste de convolucionarla con la señal original. Por otro lado, una zona muy ancha implicaría considerar una extensión muy grande de la señal en el cómputo de cada valor de la señal de salida, lo que repercutiría en la pérdida total de los valores locales.

Por esta razón es por lo que no suele usarse un filtro paso bajo ideal, sino que se usan filtros $H(w)$ con un salto menos abrupto en la frecuencia, como el de la figura 4.3c, ya que eso produce una $h(x)$ con una zona más estrecha de valores no nulos (figura 4.3d). Sin embargo, suavizar demasiado el salto en la frecuencia produce un ensanchamiento de la zona de valores significativos en $H(w)$, con lo cual dejaría de ser un filtro paso-bajo puro, ya que permitiríamos el paso de frecuencias cada vez más altas. Por tanto, debe llegarse a un compromiso entre ambos parámetros (el ancho de $H(w)$ y el de $h(x)$).

4.1.1 Filtro gaussiano

El filtro $H(w)$ que mejor optimiza este compromiso [LIN94] consiste en una función gaussiana, cuya expresión viene dada por

$$H_G(w) = e^{-\frac{1}{2}w^2\sigma^2}$$

La antitransformada de dicho filtro es también una nueva gaussiana, dada por la expresión

$$h_G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

Puede observarse cómo la diferencia fundamental entre ambas gaussianas es, además del factor de escala que multiplica a la exponencial, que la desviación estándar de una función es la inversa de la otra. En la figura 4.4 pueden verse varios filtros gaussianos con distintos valores de desviación, tanto en el dominio frecuencial como en el espacial. Puede verse claramente cómo el hecho de estrechar la zona de valores significativos en el dominio frecuencial produce un ensanchamiento en el dominio espacial, y viceversa.

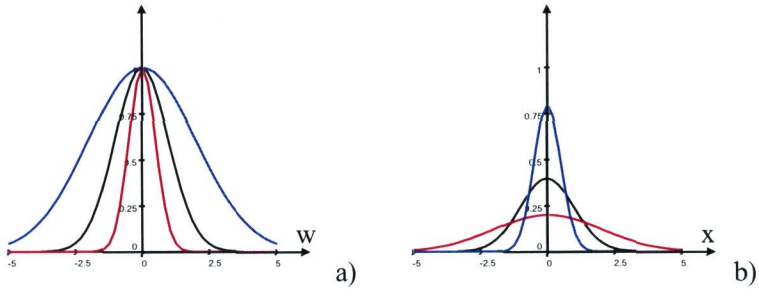


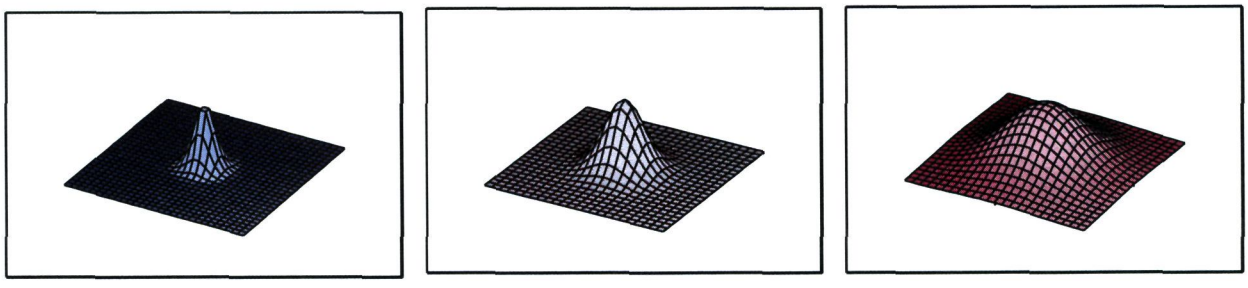
Figura 4.4: Filtros gaussianos con $\sigma = 0.5$ (en azul), 1 (en negro) y 2 (en rojo): a) dominio frecuencial. b) dominio espacial.

Volviendo a la figura 4.2, las gráficas c y d fueron obtenidas suavizando con gaussianas de diferentes desviaciones.

Filtro gaussiano bidimensional Ya que las imágenes son señales bidimensionales, puede rehacerse todo el desarrollo anterior para demostrar, de igual manera, que el filtro gaussiano es el óptimo en cuanto al compromiso de mantener limitado el ancho en ambos dominios (frecuencial y espacial). Para señales 2D, la expresión de la gaussiana pasa a ser

$$h_G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

En la siguiente figura se muestran tres gaussianas en el dominio espacial con los mismos valores de desviación usados en la figura unidimensional anterior.



4.1.2 Filtrado discreto

Para poder convolucionar una imagen con un filtro paso-bajo habrá que discretizar el filtro primero. Para ello se crea una subimagen o máscara cuyos valores sean la evaluación de la expresión del filtro en determinados valores (x, y) alrededor del punto $(0, 0)$. Por ejemplo, la máscara para $\sigma = 1.00$ tendría los siguientes valores:

$$h_G(x,y)_{\sigma=1.0} \simeq \frac{1}{628}\begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 2 & 8 & 14 & 8 & 2 & 0 \\ 1 & 8 & 37 & 61 & 37 & 8 & 1 \\ 1 & 14 & 61 & 100 & 61 & 14 & 1 \\ 1 & 8 & 37 & 61 & 37 & 8 & 1 \\ 0 & 2 & 8 & 14 & 8 & 2 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

La dimensión de esta máscara es 7×7 porque los valores más alejados son inferiores a un umbral prefijado (en este caso, un 1% del valor en el punto $(0, 0)$). Cuanto mayor sea la desviación σ, mayor será la máscara que habrá que emplear. Y de forma análoga, cuanto menor sea σ, menor tamaño tendrá la máscara. Por ejemplo, para $\sigma = 1/2$ y un umbral del 1%, la máscara es sólo de 3×3, como se muestra a continuación:

$$h_G(x,y)_{\sigma=0.5} \simeq \frac{1}{164}\begin{pmatrix} 2 & 14 & 2 \\ 14 & 100 & 14 \\ 2 & 14 & 2 \end{pmatrix} \tag{4.1}$$

En la figura 4.5 puede verse una imagen que ha sido suavizada con gaussianas de diferente valor de σ. Se observa que a mayor valor de σ, el suavizado es mayor, y por tanto el difuminado de los bordes de la imagen. Por lo tanto, con vistas a aplicar posteriormente un método detector de bordes, no debería usarse una gaussiana demasiado amplia.

Para poder contemplar mejor el efecto de disminución del ruido, en la figura 4.6 se muestra una ampliación de una pequeña zona de la imagen, y una representación tridimensional de dicha zona, donde cada pixel se ha representado por un prisma de base cuadrada, cuya altura es proporcional a la intensidad de dicho pixel. Puede verse cómo la función intensidad tiene un aspecto bastante más liso, pero a la vez puede observarse como la rampa que caracteriza al borde también se ha visto ensanchada, lo que produce un borde visual más difuminado. Éste es el mayor de los inconvenientes del suavizado de este tipo.

4.1.3 Detección del gradiente

Una vez la imagen está suavizada, ya se puede aplicar un esquema de cálculo del vector gradiente, como los que vimos en la sección 1.1, convolucionando por ejemplo con máscaras $H_x(\alpha)$ y $H_y(\alpha)$ para obtener las parciales en las direcciones x e y respectivamente, y luego obteniendo el valor del vector gradiente a partir de ellas. En la figura 4.7 se ve la diferencia entre aplicar el gradiente directamente o suavizando primero la imagen. Vemos que los valores obtenidos en el gradiente de la imagen suavizada son más regulares que en la imagen original, pero se ve que también hay un cambio en los valores. Concretamente, la magnitud del gradiente es ahora inferior (la intensidad sobre el contorno en la imagen de gradientes es menor).

Ambos procesos (suavizado y detección del gradiente) pueden combinarse en una única convolución. Sea $f(x, y)$ la imagen original. Filtrar con una gaussiana $h_G(x, y)$ y posteriormente convolucionar con la máscara $H_x(\alpha)$ para obtener la parcial en x puede expresarse como sigue

$$g(x,y) = f(x,y) * h_G(x,y) * H_x(x,y) = f(x,y) * H_{Gx}(x,y)$$

donde

$$H_{Gx}(x,y) = h_G(x,y) * H_x(x,y)$$

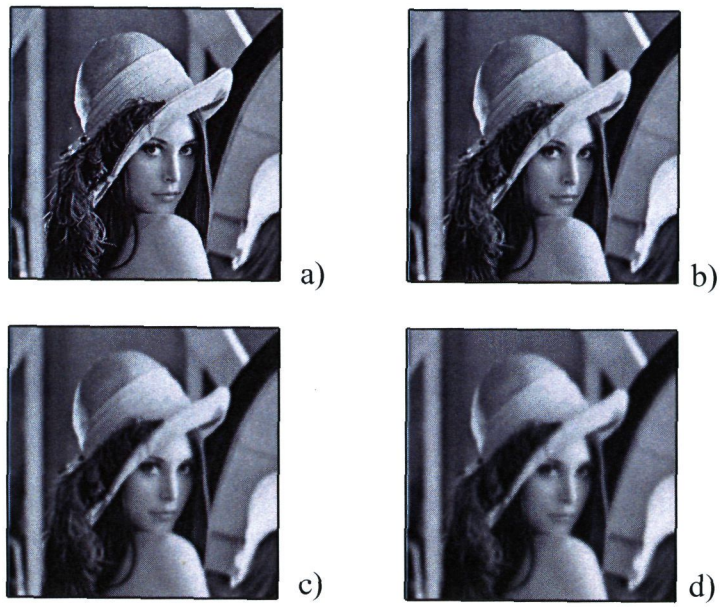


Figura 4.5: Imagen original (a) y suavizada con gaussianas de diferente σ: 1 (b), 5 (c) y 10 (d).

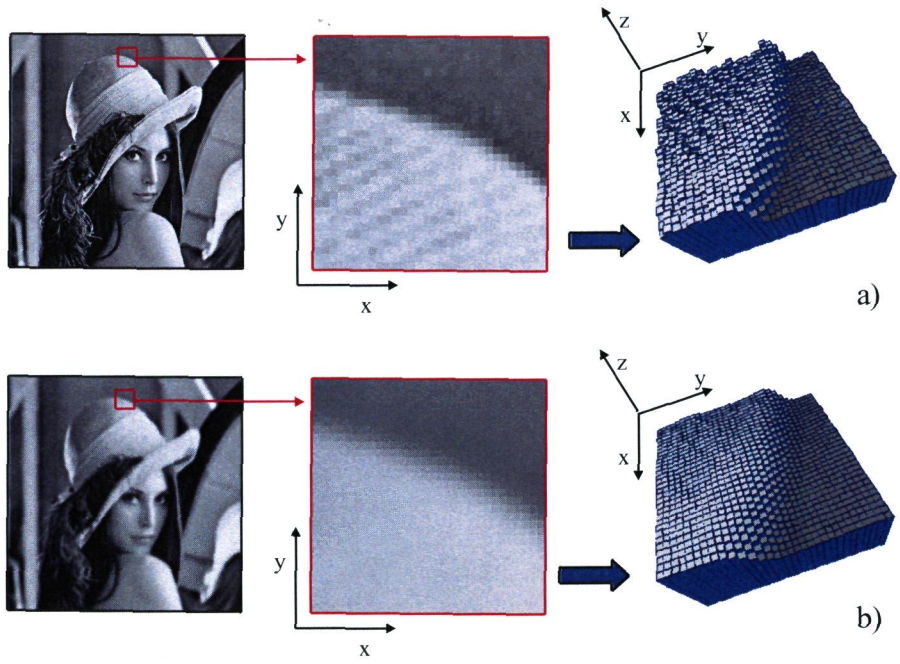


Figura 4.6: Ampliación de una zona con borde en la imagen original (a) y en la suavizada (b).

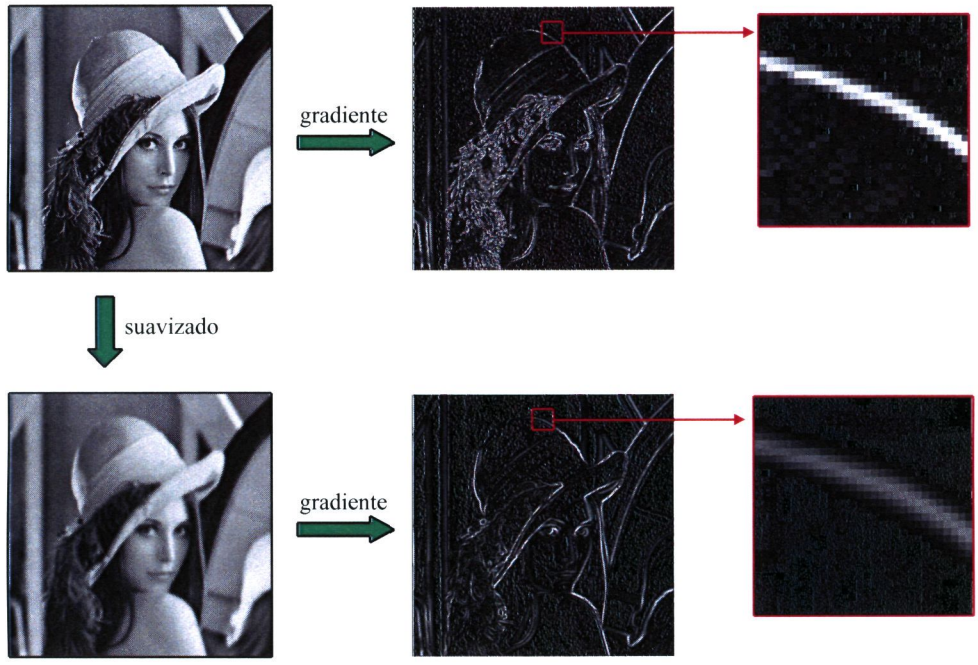


Figura 4.7: Cálculo del gradiente en la imagen original y en la suavizada

que es igual a la convolución de una máscara gaussiana con una máscara de parciales en x. Por ejemplo, suavizar con una gaussiana de $\sigma = 0.5$ (ecuación 4.1) y luego evaluar la parcial en x usando una máscara de Prewitt (ecuación 1.4) nos daría la máscara

$$\begin{aligned} H_{Gx}(x,y) &= \frac{1}{164}\begin{pmatrix} 2 & 14 & 2 \\ 14 & 100 & 14 \\ 2 & 14 & 2 \end{pmatrix} * \frac{1}{6}\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} = \\ &= \frac{1}{972}\begin{pmatrix} -2 & -14 & 0 & 14 & 2 \\ -16 & -114 & 0 & 114 & 16 \\ -18 & -128 & 0 & 128 & 18 \\ -16 & -114 & 0 & 114 & 16 \\ -2 & -14 & 0 & 14 & 2 \end{pmatrix} \end{aligned}$$

Convolucionar con esta máscara es equivalente a hacer los dos procesos por separado.

4.1.4 Error cometido por el proceso de suavizado previo

En la sección 1.3 se demostró que la técnica de evaluar el gradiente por medio de las máscaras H_x y H_y no funcionaba de forma exacta, ni siquiera en imágenes ideales. Lógicamente, si a esa imagen se le aplica un filtro gaussiano para suavizar, el resultado será incluso más inexacto, y más aún considerando que al aplicar una convolución discreta se comete un error con respecto a la convolución de la gaussiana en el plano continuo.

Para ver un caso concreto, tomemos el siguiente ejemplo: Sea F una imagen ideal con un contorno de primer orden, de pendiente 1/2, que separa dos zonas de intensidad 0 y 100 a cada lado del borde (ver figura 1.18).

		Módulo		Dirección		Desplazamiento	
Método	**Imagen**	Valor	Error	Valor	Error	Valor	Error
Tradicional	F	97.54	2.46%	19.98	6.59^o	NC	–
Tradicional	$G_{0.5}$	86.60	13.40%	21.84	4.73^o	NC	–
Tradicional	G_1	57.14	42.86%	24.39	2.18^o	NC	–
Propuesto	F	100	0.00%	26.57	0.00^o	−0.25	0%
Propuesto	$G_{0.5}$	99.87	0.13%	26.60	0.03^o	−0.29	4%
Propuesto	G_1	87.15	12.85%	26.89	0.32^o	−0.39	14%

Tabla 4.1: Valores obtenidos por cada método en cada imagen al obtener el contorno de un borde de pendiente 1/2. F: imagen ideal. G: imagen suavizada. NC: no lo calcula

Los valores para los pixels son:

$$F = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 25 & 75 \\ 0 & 0 & 0 & 0 & 0 & 25 & 75 & 100 & 100 \\ 0 & 0 & 0 & 25 & 75 & 100 & 100 & 100 & 100 \\ 0 & 25 & 75 & 100 & 100 & 100 & 100 & 100 & 100 \\ 75 & 100 & 100 & 100 & 100 & 100 & 100 & 100 & 100 \\ 100 & 100 & 100 & 100 & 100 & 100 & 100 & 100 & 100 \end{pmatrix}$$

Centrémonos en el pixel central, de valor 75. Si tomamos como origen de coordenadas el centro geométrico de dicho pixel, el contorno pasa exactamente $0.25h$ unidades por encima del dicho origen. La pendiente de dicho contorno es aproximadamente 26.57 grados.

En la tabla 4.1 se muestran los resultados de usar nuestro método propuesto del capítulo anterior y el método tradicional para obtener los parámetros del contorno, tanto sobre la imagen original como sobre las imágenes resultantes de convolucionar con una gaussiana de desviación 0.5 ($G_{0.5}$) y otra de desviación 1.0 (G_1).

Como era de prever, para la imagen original F nuestro método logra una exactitud total, mientras que el método tradicional comete un error del 2.5% en la magnitud del salto de intensidad y de 6.5 grados en el valor de la orientación. Para las imágenes suavizadas, el método tradicional va mejorando el valor de la orientación, llegando a ser de 2.1 grados en G_1, pero va empeorando el valor del salto drásticamente. Nuestro método es más estable, pero también tiene un error, llegando a ser del 13% en el valor del salto, y de 0.32 grados en la orientación.

Por tanto, vamos a desarrollar a continuación un esquema que siga siendo exacto cuando la imagen ha sido suavizada. Abordaremos primero el problema en 1 dimensión, y posteriormente lo haremos para dimensión 2.

4.2 Método suavizado de primer orden en imágenes 1D

Llamaremos imagen 1D a una imagen compuesta por una única fila. En este tipo de imágenes, sólo tiene sentido hablar de contornos verticales. Una imagen ideal 1D con un contorno vertical es una imagen cuyos valores son

$$F = (\ B \quad \ldots \quad B \quad C \quad A \quad \ldots \quad A\ ) \tag{4.2}$$

donde

$$C = \frac{t}{h}B + \frac{h-t}{h}A$$

Una imagen de este tipo puede verse en la figura 4.8, donde aparece un contorno vertical situado en el pixel central, a una distancia t de su margen izquierdo, separando dos zonas de intensidad A y B a cada lado del

contorno. El interés en estas imágenes radica solamente en conocer con exactitud el salto de intensidad, $A - B$, y la posición sub-pixel, t, pero no así su orientación pues siempre será vertical.

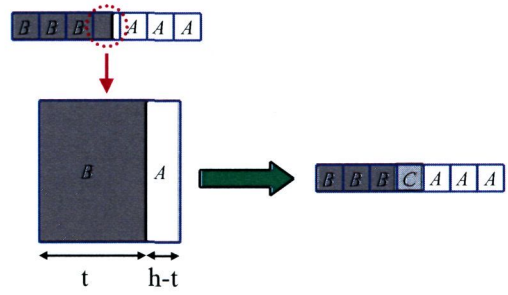


Figura 4.8: Imagen 1D con un contorno vertical en el pixel central.

Para aplicar un suavizado hay que convolucionar con una máscara como la siguiente

$$H_n = \begin{pmatrix} a_n & a_{n-1} & \dots & a_1 & a_0 & a_1 & \dots & a_{n-1} & a_n \end{pmatrix} \tag{4.3}$$

donde se cumplen las siguientes 2 propiedades:

$$\begin{aligned} a_0 &\geq a_1 \geq \dots \geq a_n > 0 \\ 1 &= a_n + a_{n-1} + \dots + a_1 + a_0 + a_1 + \dots + a_{n-1} + a_n \end{aligned} \tag{4.4}$$

A dicha máscara la llamaremos **máscara de suavizado de radio** n, por tener n valores no nulos a cada lado del pixel central. La primera propiedad es consecuencia del perfil típico de un filtro paso bajo, tal y como se vio en la sección anterior. La segunda propiedad es para normalizar la máscara, y que la convolución no afecte a la intensidad media de la imagen.

El resultado de la convolución producirá la imagen $G = F * H_n$, y estamos interesados en obtener los parámetros del contorno existente en F a partir de los pixels de la imagen G. Veamos primero el caso más sencillo.

4.2.1 Convolución con una máscara de radio 1

Una máscara de radio 1 viene definida como

$$H_1 = \begin{pmatrix} a_1 & a_0 & a_1 \end{pmatrix}$$

donde

$$\begin{aligned} a_0 &\geq a_1 > 0 \\ 1 &= a_0 + 2a_1 \end{aligned}$$

La imagen G suavizada será de la forma siguiente:

$$G = \begin{pmatrix} B & \dots & B & G_{i-1} & G_i & G_{i+1} & A & \dots & A \end{pmatrix}$$

donde

$$\begin{aligned} G_{i-1} &= (a_1 + a_0)B + a_1 C \\ G_i &= a_1 B + a_0 C + a_1 A \\ G_{i+1} &= a_1 C + (a_0 + a_1)A \end{aligned}$$

Como puede verse, si en la imagen original F existía un único pixel, al que se ha llamado i, con intensidad diferente de A y B que representaba el borde en la imagen, en la imagen suavizada G son tres los pixels con intensidad intermedia (ver figura 4.9a y b). Esto es debido al efecto de difuminado característico de estos filtros, tal y como comentábamos en la sección anterior.

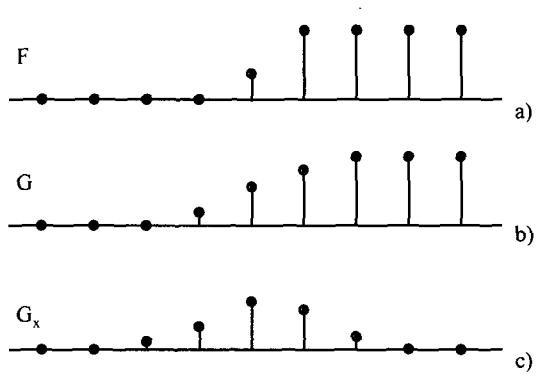


Figura 4.9: a) Imagen 1D ideal con borde en el pixel central. b) Imagen suavizada con una máscara de radio 1. c) Imagen de derivadas de la imagen suavizada.

Elección del pixel borde correcto Lo primero que debe hacerse es encontrar el pixel candidato a ser detectado como pixel borde. En la sección 2.1.1 se vio que en una imagen ideal, un borde viene caracterizado por un perfil típico en la imagen de parciales. Es decir, evaluando la imagen de derivadas en x de la imagen F obtendríamos la imagen

$$F_x = \frac{1}{2h}\left(0 \quad \ldots \quad 0 \quad C-B \quad A-B \quad A-C \quad 0 \quad \ldots \quad 0 \right)$$

donde se cumple que

$$0 \leq C - B \leq A - B \geq A - C \geq 0$$

Esto indicaba que el pixel con mayor derivada era aquél por el que pasaba el contorno. ¿Seguirá existiendo un perfil similar en la imagen de derivadas de G? La respuesta es sí. Basta ver que la imagen de derivadas de G es igual a

$$G_x = \frac{1}{2h}\left(0 \quad \ldots \quad 0 \quad G_{i-1}-B \quad G_i - B \quad G_{i+1} - G_{i-1} \quad A - G_i \quad A - G_{i+1} \quad 0 \quad \ldots \quad 0 \right)$$

donde se cumple que

$$0 \leq G_{i-1} - B \leq G_i - B \leq G_{i+1} - G_{i-1} \geq A - G_i \geq A - G_{i+1} \geq 0$$

Para demostrarlo basta ver que

$$
\begin{aligned}
G_{i-1} - B &= \left(\frac{A-B}{2h}\right)\frac{a_1(h-t)}{h} \geq 0 \\
(G_i - B) - (G_{i-1} - B) &= \frac{A-B}{2h^2}(a_1 h + (h-t)(1-3a_1)) \geq 0 \\
(G_{i+1} - G_{i-1}) - (G_i - B) &= \frac{A-B}{2h^2} t a_0 \geq 0 \\
(G_{i+1} - G_{i-1}) - (A - G_i) &= \frac{A-B}{2h^2}(h-t)a_0 \geq 0 \\
(A - G_i) - (A - G_{i+1}) &= \frac{A-B}{2h^2}(a_1 h + t(1-3a_1)) \geq 0 \\
A - G_{i+1} &= \left(\frac{A-B}{2h}\right)\frac{a_1 t}{h} \geq 0
\end{aligned}
$$

ya que sabemos que $h \geq t \geq 0$ y $a_1 \leq 1/3$. Este última condición es necesaria, ya que en caso contrario se cumpliría que $a_0 < a_1$, lo cual sería ilógico en un filtro paso bajo, puesto que el pixel central no puede tener menor peso que un pixel vecino en la máscara de convolución. Por lo tanto, queda demostrado que el pixel con valor máximo en la imagen de derivadas G_x indica el pixel de la imagen original por la que pasa el contorno (ver figura 4.9c).

Cálculo de la magnitud del salto de intensidad Sea i el pixel que hemos etiquetado como borde, y llamemos por simplicidad X a la imagen de derivadas G_x. Para encontrar el salto de intensidad, sólo hay que ver que

$$X_{i-1} + X_{i+1} = \frac{G_i - B}{2h} + \frac{A - G_i}{2h} = \frac{A-B}{2h} \tag{4.5}$$

con lo cual podemos estimar el salto de intensidad, S, como

$$S = 2\,(X_{i-1} + X_{i+1})$$

Cálculo de la posición del contorno Si para conocer el salto utilizamos la suma de los pixels vecinos en la imagen X, para obtener la posición usaremos la diferencia

$$
\begin{aligned}
X_{i+1} - X_{i-1} &= \frac{A - G_i}{2h} - \frac{G_i - B}{2h} = \frac{A+B}{2h} - \frac{G_i}{h} = \\
&= \frac{A+B}{2h} - \frac{a_1 B + a_0 C + a_1 A}{h} = \\
&= \frac{A}{2h}\left(1 - 2a_1 - 2a_0 + 2a_0\frac{t}{h}\right) + \frac{B}{2h}\left(1 - 2a_1 - 2a_0\frac{t}{h}\right) = \\
&= \left(\frac{A-B}{h}\right) a_0 \frac{t - h/2}{h} = \left(\frac{A-B}{h}\right) a_0 d
\end{aligned}
$$

Vemos que el resultado es el producto de tres factores: el primero es el salto de intensidad, que ya lo hemos calculado. El segundo es el valor central de la máscara. Finalmente el tercer factor es precisamente el valor que queremos calcular, ya que d representa el desplazamiento horizontal entre el contorno y la vertical central del pixel, relativo a la longitud del pixel, $-1/2 \leq d \leq 1/2$ (ver figura 4.8).

Para poder despejar d se hace por tanto necesario conocer el valor de a_0. Podemos asumir que este valor es conocido, y en caso contrario, deducirlo a partir del valor del pixel central, X_i, ya que

$$X_i = \frac{G_{i+1} - G_{i-1}}{2h} = \frac{A-B}{2h}(a_0 + a_1) = \frac{A-B}{2h}(1 - a_1)$$

De esta expresión, y utilizando la ecuación 4.5, podemos obtener primero el valor de a_1

$$a_1 = 1 - \frac{X_i}{X_{i-1} + X_{i+1}}$$

y luego el de $a_0 = 1 - 2a_1$.
Con todo esto ya podemos enunciar el siguiente lema:

Lema 4.1 *Sea F una imagen unidimensional con un único borde en uno de sus pixels, que ha sido suavizada con una máscara de suavizado de radio 1, $H_1 = (\ a_1 \quad a_0 \quad a_1\ )$, dando como resultado la imagen G. A partir de los datos de la imagen suavizada es posible obtener los parámetros del contorno de la imagen original (salto de intensidad y desplazamiento), realizando los siguientes pasos:*
a) se evalúa la imagen de derivadas de G usando la fórmula básica

$$X_k = \frac{1}{2h}(G_{k+1} - G_{k-1})$$

b) se obtiene el pixel borde, i, buscando el pixel cuya X_i sea máxima
c) el salto de intensidad, S, se obtiene haciendo

$$S = 2(X_{i-1} + X_{i+1})$$

d) en caso de no conocer los valores de la máscara de antemano, se obtienen haciendo

$$\begin{aligned} a_1 &= 1 - \frac{2X_i}{S} \\ a_0 &= 1 - 2a_1 \end{aligned}$$

e) el desplazamiento con respecto a la vertical central se obtiene haciendo

$$d = \frac{X_{i+1} - X_{i-1}}{a_0 S}$$

Por tanto, aplicando este lema a una imagen suavizada, incluso sin conocer los valores de la máscara empleados (aunque sí hay que saber que ésta era de radio 1), podemos recuperar con exactitud los parámetros del borde original. Veamos ahora el caso para una máscara de radio n.

4.2.2 Convolución con una máscara de radio n

Una máscara de radio n viene definida según la ecuación 4.3, donde se cumplen las propiedades dadas en las ecuaciones 4.4. La imagen G suavizada será de la forma siguiente:

$$G = (\ B \ \ldots \ B \ G_{i-n} \ \ldots \ G_{i-1} \ G_i \ G_{i+1} \ \ldots \ G_{i+n} \ A \ \ldots \ A\ )$$

donde

$$G_k = \sum_{m=-n}^{n} a_m F_{k+m}$$

y donde se ha supuesto que $a_{-k} = a_k$.

Como puede verse, ahora en la imagen suavizada G aparece una franja de $2n+1$ pixels cuya intensidad se encuentra entre los valores de A y B, teniendo por tanto una difuminación mayor, proporcional al radio n de la máscara empleada (ver figura 4.10b).

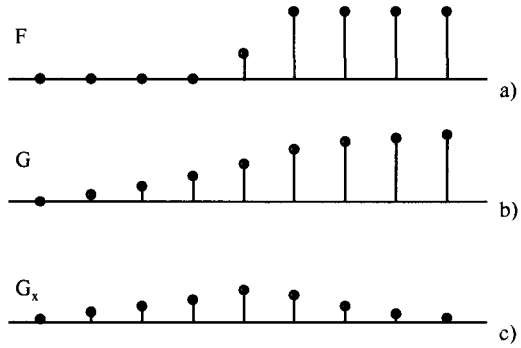


Figura 4.10: a) Imagen 1D ideal con borde en el pixel central. b) Imagen suavizada con una máscara de radio 3. c) Imagen de derivadas de la imagen suavizada.

Para elegir el pixel borde correcto, de nuevo puede demostrarse que la imagen de derivadas de G, dada por

$$G_x = (\ 0\ \ ...\ \ 0\ \ X_{i-(n+1)}\ \ ...\ \ X_{i-1}\ \ X_i\ \ X_{i+1}\ \ ...\ \ X_{i+(n+1)}\ \ 0\ \ ...\ \ 0\ )$$

cumple que

$$0 \leq X_{i-(n+1)} \leq ... \leq X_{i-1} \leq X_i \geq X_{i+1} \geq ... \geq X_{i+(n+1)} \geq 0$$

Esto significa que se sigue cumpliendo que el pixel con valor máximo en la imagen de derivadas G_x indica el pixel de la imagen original por la que pasa el contorno (ver figura 4.10c).

Cálculo de los parámetros del contorno Al igual que hicimos antes, hallemos la suma de las derivadas de los pixels vecinos al pixel i :

$$X_{i-1} + X_{i+1} = \frac{1}{2h}(G_{i+2} - G_{i-2})$$

Sabemos que

$$\begin{aligned} G_{i+2} &= a_nB + ... + a_3B + a_2C + a_1A + a_0A + ... + a_nA \\ G_{i-2} &= a_nB + ... + a_0B + a_1B + a_2C + a_3A + ... + a_nA \end{aligned}$$

Por tanto

$$X_{i-1} + X_{i+1} = \frac{A-B}{2h}(a_0 + 2a_1 + a_2)$$

Esto significa que con la suma de las derivadas de los vecinos seguimos pudiendo estimar el salto de intensidad, aunque en este caso sí que es preciso conocer los valores de la máscara.

Calculando ahora la diferencia de las derivadas de los vecinos, obtenemos que

$$X_{i+1} - X_{i-1} = \frac{1}{2h}(G_{i+2} - 2G_i + G_{i-2})$$

Sabiendo que

$$G_i = a_nB + ... + a_1B + a_0C + a_1A + ... + a_nA$$

obtenemos que

$$X_{i+1} - X_{i-1} = \frac{a_0 - a_2}{2h}(A + B - 2C) = \left(\frac{A-B}{h}\right)(a_0 - a_2)\frac{t - h/2}{h} = \\ = \left(\frac{A-B}{h}\right)(a_0 - a_2)\, d$$

Al igual que para radio 1, obtenemos una expresión de tres factores, donde el factor d es el que queremos calcular. Vemos que también aquí es preciso conocer los valores de la máscara usada para obtener G. Con lo cual ya estamos en disposición de enunciar el siguiente lema:

Lema 4.2 *Sea F una imagen unidimensional con un único borde en uno de sus pixels, que ha sido suavizada con una máscara de suavizado de radio n, $H_n = (\ a_n \ \ldots \ a_1 \ a_0 \ a_1 \ \ldots \ a_n\ )$, donde al menos son conocidos los valores a_0, a_1 y a_2, dando como resultado la imagen G. A partir de los datos de la imagen suavizada es posible obtener los parámetros del contorno de la imagen original (salto de intensidad y desplazamiento), realizando los siguientes pasos:*

a) se evalúa la imagen de derivadas de G usando la fórmula básica

$$X_k = \frac{1}{2h}(G_{k+1} - G_{k-1})$$

b) se obtiene el pixel borde, i, buscando el pixel cuya X_i sea máxima

c) el salto de intensidad, S, se obtiene haciendo

$$S = \frac{2}{a_0 + 2a_1 + a_2}(X_{i-1} + X_{i+1})$$

d) el desplazamiento con respecto a la vertical central se obtiene haciendo

$$d = \frac{X_{i+1} - X_{i-1}}{(a_0 - a_2)\, S}$$

Vemos que para máscaras de radio mayor que 1, es preciso conocer los valores de la máscara para poder deducir los parámetros del contorno en la imagen original. Aunque no hacen falta conocer todos: sólamente los tres valores más significativos. El resto de valores es irrelevante, incluido el valor del propio radio de la máscara, n. Esto es así porque, atendiendo solamente a una pequeña vecindad alrededor del pixel en cuestión en la imagen suavizada G, es imposible determinar con exactitud los parámetros del contorno en la imagen original F, si no se conocen los coeficientes de la máscara, ya que diferentes imagenes F podrían producir imágenes suavizadas G tales que la vecindad del pixel en cuestión coincidiera en ambas.

A continuación pasaremos a realizar el método para el caso bidimensional.

4.3 Método suavizado de primer orden en imágenes 2D

Un suavizado 2D se hace convolucionando la imagen con una máscara como la siguiente

$$H_n = \begin{pmatrix} a_{nn} & \ldots & a_{1n} & a_{0n} & a_{1n} & \ldots & a_{nn} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{1n} & \ldots & a_{11} & a_{01} & a_{11} & \ldots & a_{1n} \\ a_{0n} & \ldots & a_{01} & a_{00} & a_{01} & \ldots & a_{0n} \\ a_{1n} & \ldots & a_{11} & a_{01} & a_{11} & \ldots & a_{1n} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ a_{nn} & \ldots & a_{1n} & a_{0n} & a_{1n} & \ldots & a_{nn} \end{pmatrix}$$

donde se cumple que

$$a_{k,k} > a_{k,k+1} > ... > a_{k,n}, \quad \forall k : 0 \leq k \leq n$$

y que la suma de todos los elementos de la máscara vale 1.

Sea ahora F una imagen ideal como la de capítulos anteriores, con un contorno de primer orden con pendiente entre 0 y 1, que separa dos zonas de intensidad A y B, y que atraviesa el pixel central de la imagen, (al que llamaremos por simplicidad $F_{0,0}$), como muestra la figura 4.11a.

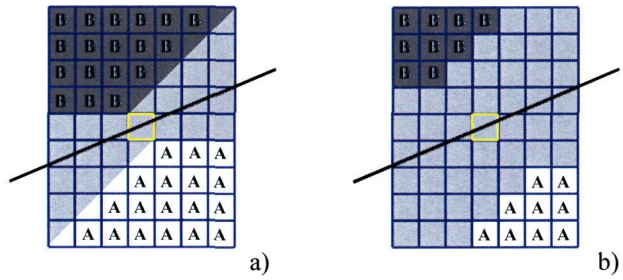


Figura 4.11: a) Imagen ideal con un contorno recto. b) Suavizado de la imagen con una máscara H_1. Los pixels en color gris claro pueden tomar valores intermedios entre A y B en función de la posición y orientación exacta del contorno.

Los pixels de dicha figura etiquetados con las intensidades A y B indican los pixels que con toda seguridad están completamente a uno o a otro lado del contorno. La zona gris clara intermedia indica todos aquellos pixels por los cuales puede pasar el contorno, y por lo tanto desconocemos a priori su intensidad. Lo que sí sabemos es que dicha intensidad está relacionada con las áreas interiores al pixel a cada lado del contorno, mediante la siguiente expresión:

$$F_{i,j} = \frac{P_{i,j}}{h^2} A + \frac{h^2 - P_{i,j}}{h^2} B$$

donde $P_{i,j}$ representa el área interior al pixel (i,j) que se encuentra bajo la recta, y h^2 es el área del pixel. Por tanto, $0 \leq P_{i,j} \leq h^2$.

Ya que la pendiente está entre 0 y 1, nos centraremos en el pixel donde la parcial en y es máxima en su columna, al igual que ocurría con el método original cuando se aplicaba en F.

4.3.1 Convolución con una máscara de radio 1

Una máscara de radio 1 tiene la siguiente expresión:

$$H_1 = \begin{pmatrix} a_{11} & a_{01} & a_{11} \\ a_{01} & a_{00} & a_{01} \\ a_{11} & a_{01} & a_{11} \end{pmatrix}$$

donde se cumple que

$$\begin{aligned} 1 &= a_{00} + 4a_{01} + 4a_{11} \\ a_{00} &\geq a_{01} \geq a_{11} > 0 \end{aligned}$$

Al convolucionar F con la máscara H_1 obtenemos la imagen G tal que

$$G = F * H_1$$

donde la intensidad de cada pixel de G viene dada por la expresión

$$G_{i,j} = a_{00}F_{i,j} + a_{01}\left(F_{i-1,j} + F_{i,j-1} + F_{i,j+1} + F_{i+1,j}\right) + a_{11}\left(F_{i-1,j-1} + F_{i-1,j+1} + F_{i+1,j-1} + F_{i+1,j+1}\right)$$

En la figura 4.11b puede verse el aspecto que tendrá la imagen G. Como puede apreciarse, la zona de posibles valores intermedios entre A y B (color gris claro en la figura) es ahora mayor que en la imagen original, debido al efecto de difuminado producido por el filtro.

Diferencias entre los casos 1D y 2D En el caso unidimensional podía conocerse a priori la expresión que toma cada uno de los pixels de una imagen ideal con un contorno (ecuación 4.2). Pero en dos dimensiones, si consideramos que la recta que estamos buscando sigue la expresión $y = a + bx$, donde el origen de coordenadas está en el centro geométrico del pixel central, $F_{0,0}$, esto no es así exactamente. Cuando desarrollamos nuestro método original en el capítulo previo vimos que, aunque no podamos relacionar la intensidad de un pixel con la expresión de la recta, sí podemos hacerlo con la suma de una columna entera de pixels. Así, la suma de los pixels de la columna i en la imagen F será:

$$M_i = \sum_{j=-m}^{m} F_{i,j} = A\frac{P_i}{h^2} + B\left(N - \frac{P_i}{h^2}\right) \tag{4.6}$$

donde $N = 2m + 1$ es el número de pixels que hemos tomado de la columna, y P_i representa el área de la columna i bajo la recta.

Para obtener una expresión para el área P_i, ya hemos visto que es sencillo siempre y cuando la altura de la columna sea tal que la recta del contorno la atraviese siempre de izquierda a derecha, pero nunca por arriba ni por abajo. Así por ejemplo, atendiendo a la figura 4.11a, para la columna central ($i = 0$) bastará con tomar $m = 1$ en la expresión 4.6, pero para la columna $i = 1$ habrá que tomar $m = 2$ como mínimo. Una vez asegurado esto, sabemos que el área P_i ha de ser igual a

$$P_i = \int_{(i-\frac{1}{2})h}^{(i+\frac{1}{2})h} \left(a + bx + \left(m + \frac{1}{2}\right)h\right) dx = ah + bih^2 + \left(m + \frac{1}{2}\right)h^2$$

Cálculo de la posición del contorno. Para obtener una primera ecuación de donde obtener posteriormente los parámetros del contorno, tomemos la suma de los pixels con valores intermedios en la columna central ($i = 0$) de la imagen G, esto es

$$\begin{aligned} S_M &= \sum_{j=-3}^{3} G_{0,j} = a_{00}\sum_{j=-3}^{3} F_{0,j} + a_{01}\sum_{j=-3}^{3}\left(F_{-1,j} + F_{0,j-1} + F_{0,j+1} + F_{1,j}\right) + \\ &+ a_{11}\sum_{j=-3}^{3}\left(F_{-1,j-1} + F_{-1,j+1} + F_{1,j-1} + F_{1,j+1}\right) \end{aligned}$$

Esta expresión puede verse de forma visual en la figura 4.12. Para simplificarla haremos lo siguiente: en primer lugar, podemos afirmar que, en las tres columnas centrales de la imagen original F, se cumple que

$$\begin{aligned} F_{i,j} &= B, \forall j \leq -3 \\ F_{i,j} &= A, \forall j \geq 3 \end{aligned}$$

En segundo lugar, llamemos M_i a la suma de los 5 pixels de la columna i, $i = \{-1, 0, 1\}$ en la imagen original, es decir,

$$M_i = \sum_{j=-2}^{2} F_{i,j}$$

Con todo ello, la suma S_M puede expresarse como

$$\begin{aligned} S_M &= a_{00}\,(B + M_0 + A) + a_{01}\,(4B + M_{-1} + 2M_0 + M_1 + 4A) + a_{11}\,(4B + 2M_{-1} + 2M_1 + 4A) = \\ &= (a_{00} + 2a_{01})(M_0 + A + B) + (a_{01} + 2a_{11})(M_{-1} + M_1 + 2(A + B)) \end{aligned} \tag{4.7}$$

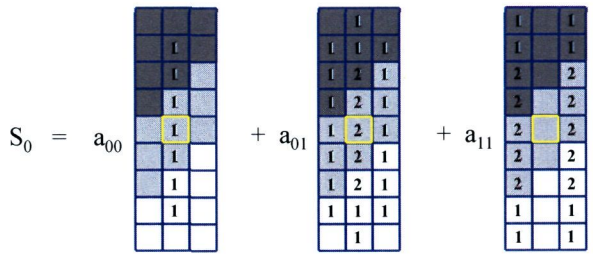


Figura 4.12: Suma de la columna central de la imagen G.

Como se vio antes en la ecuación 4.6, la expresión para M_i en este caso es la siguiente:

$$M_i = A\frac{P_i}{h^2} + B\left(5 - \frac{P_i}{h^2}\right) \tag{4.8}$$

donde P_i es como sigue:

$$P_i = ah + bih^2 + \frac{5}{2}h^2$$

En la figura 4.11a puede verse que la condición de que el contorno cruce la columna de izquierda a derecha se cumple para las tres columnas centrales de la imagen F que estamos tratando. Combinando entonces las expresiones de M_i y P_i con la de S_M, podemos obtener la expresión final para esta suma:

$$S_M = \left(\frac{a}{h}(A - B) + \frac{7}{2}(A + B)\right)(a_{00} + 4a_{01} + 4a_{11}) = \frac{a}{h}(A - B) + \frac{7}{2}(A + B)$$

De aquí ya podemos obtener la expresión final para el parámetro a como sigue:

$$a = \frac{2S_M - 7(A + B)}{2(A - B)}h$$

Falta conocer los valores de A, B y h. Para h escogemos el valor 1 como en capítulos anteriores. Para estimar A y B, habrá que hacer como en el método original, es decir, irse a aquellos pixels lo más cercanos al pixel central donde sepamos que la intensidad es A o B, pero no intermedia. La diferencia es que ahora habrá que alejarse un poco más. Por ejemplo, atendiendo a la figura 4.11b podemos usar las siguientes expresiones

$$\begin{aligned} B &= \frac{G_{-1,-3} + G_{-1,-4} + G_{0,-4}}{3} \\ A &= \frac{G_{0,4} + G_{1,4} + G_{1,3}}{3} \end{aligned}$$

Podría haberse tomado un único pixel para estimar A y otro para B en lugar de usar tres pixels para cada uno. Sin embargo se ha hecho así porque, aunque en un caso ideal obtendríamos el mismo valor, en un caso real es preferible tomar un valor promediado en lugar de un único pixel.

Cálculo de la orientación. Para estimar la pendiente de la recta b, podemos proceder de forma similar pero con las columnas izquierda y derecha del pixel central, teniendo en cuenta ahora que los 5 valores de la imagen G con intensidad intermedia en esas columnas están desplazados un pixel arriba (para la de la derecha) y un pixel abajo (para la de la izquierda).

Por tanto, sean las sumas S_L y S_R como siguen:

$$\begin{aligned} S_L &= \sum_{j=-2}^{4} G_{-1,j} = (a_{00}+2a_{01})(L_{-1}+A+B)+(a_{01}+2a_{11})(L_{-2}+L_0+2\,(A+B)) \\ S_R &= \sum_{j=-4}^{2} G_{1,j} = (a_{00}+2a_{01})(R_1+A+B)+(a_{01}+2a_{11})(R_0+R_2+2\,(A+B)) \end{aligned} \tag{4.9}$$

donde

$$\begin{aligned} L_i &= \sum_{j=-1}^{3} F_{i,j} = A\frac{Q_i}{h^2}+B\left(5-\frac{Q_i}{h^2}\right), \forall i=\{-2,-1,0\} \\ R_i &= \sum_{j=-3}^{1} F_{i,j} = A\frac{T_i}{h^2}+B\left(5-\frac{T_i}{h^2}\right), \forall i=\{0,1,2\} \end{aligned} \tag{4.10}$$

y donde

$$\begin{aligned} Q_i &= \int_{(i-\frac{1}{2})h}^{(i+\frac{1}{2})h}\left(a+bx+\frac{7}{2}h\right)dx = ah+bih^2+\frac{7}{2}h^2 \\ T_i &= \int_{(i-\frac{1}{2})h}^{(i+\frac{1}{2})h}\left(a+bx+\frac{3}{2}h\right)dx = ah+bih^2+\frac{3}{2}h^2 \end{aligned}$$

y donde también se ha supuesto que en las columnas $i=\{-2,-1,0\}$ se cumple que

$$\begin{aligned} F_{i,j} &= B, \forall j \le -2 \\ F_{i,j} &= A, \forall j \ge 4 \end{aligned}$$

y que en las columnas $i=\{0,1,2\}$ se cumple que

$$\begin{aligned} F_{i,j} &= B, \forall j \le -4 \\ F_{i,j} &= A, \forall j \ge 2 \end{aligned}$$

Con esto se llegan a las expresiones para S_L y S_R, que son como siguen:

$$\begin{aligned} S_L &= \left(\frac{a}{h}-b\right)(A-B)+\frac{9}{2}A+\frac{5}{2}B \\ S_R &= \left(\frac{a}{h}+b\right)(A-B)+\frac{5}{2}A+\frac{9}{2}B \end{aligned}$$

Si realizamos la diferencia de ambas obtenemos

$$S_R - S_L = 2\,(A - B)\,(b - 1)$$

de donde podemos despejar la expresión para la pendiente de la recta, b :

$$b = 1 + \frac{S_R - S_L}{2(A - B)} \tag{4.11}$$

Con esto damos pie al siguiente lema

Lema 4.3 *Sea F una imagen ideal con un contorno recto de pendiente entre 0 y 1, que atraviesa el pixel (i, j), y que ha sido suavizada con una máscara de suavizado de radio 1, dando como resultado la imagen G. A partir de los datos de la imagen suavizada G es posible obtener los parámetros del contorno de la imagen original F (salto de intensidad, desplazamiento y orientación), realizando los siguientes pasos:*

a) obtenemos las sumas de las columnas izquierda, central y derecha de la siguiente manera:

$$\begin{aligned} S_L &= \sum_{k=-2}^{4} G_{i-1,j+k} \\ S_M &= \sum_{k=-3}^{3} G_{i,j+k} \\ S_R &= \sum_{k=-4}^{2} G_{i+1,j+k} \end{aligned}$$

b) obtenemos también una estimación para los valores de intensidad a ambos lados del borde

$$\begin{aligned} B &= \frac{G_{i-1,j-3} + G_{i-1,j-4} + G_{i,j-4}}{3} \\ A &= \frac{G_{i,j+4} + G_{i+1,j+4} + G_{i+1,j+3}}{3} \end{aligned}$$

c) a continuación detectamos los coeficientes de la recta $y = a + bx$ como sigue:

$$\begin{aligned} a &= \frac{2S_M - 7(A + B)}{2(A - B)} \\ b &= 1 + \frac{S_R - S_L}{2(A - B)} \end{aligned}$$

d) finalmente, calculamos los parámetros del contorno

$$\begin{aligned} \text{corte con la vertical central del pixel} &: \quad (0, a) \\ \text{vector normal} &: \quad N = \frac{A - B}{\sqrt{1 + b^2}} \left[\; b, \quad -1 \;\right] \end{aligned}$$

Con este lema queda demostrado que, usando exclusivamente la información de la imagen suavizada G, independientemente de los valores a_{ij} usados en la máscara de suavizado (aunque eso sí, sabiendo que la

máscara era de radio 1), pueden obtenerse de forma exacta los parámetros del contorno. A diferencia del método original, y debido al efecto de difuminado provocado por la convolución, la información de vecindad de cada pixel necesaria para aplicar el método es la incluida en una ventana 9×3 centrada en el pixel, la cual es un poco más alargada que en el método anterior (ver figura 4.13).

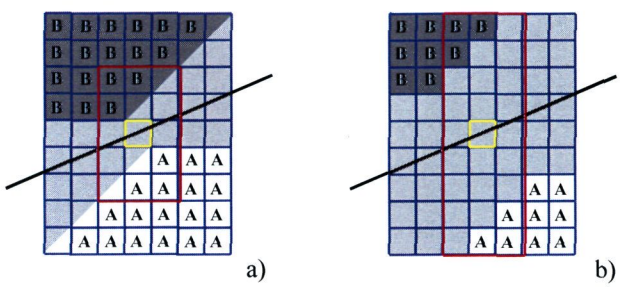


Figura 4.13: a) Ventana usada por el método propuesto inicial sobre la imagen origina F. b) Ventana usada por el método propuesto suavizado sobre la imagen suavizada G.

Como el método es válido para cualquier máscara de suavizado de radio 1, esto incluye a la máscara

$$H_1 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Esta máscara produce de nuevo la imagen original, con lo cual podemos afirmar que este método seguiría funcionando si lo aplicamos directamente en la imagen original F. Sin embargo, hay que decir que en los casos en que F no tenga ruido, es más conveniente aplicar el método desarrollado en capítulos anteriores, pues la información que se usa para estimar el contorno en cada pixel es más local (una vecindad más reducida).

4.3.2 Convolución con una máscara de radio n

Para máscaras de radio mayor, la zona difuminada del borde es cada vez más amplia. Por ejemplo, en la figura 4.14 puede verse una imagen con borde F suavizada con una máscara de radio 1 y con otra de radio 2. En esta última, para acceder a pixels con los cuales poder dar una estimación del valor de A o de B habría que alejarse en exceso del pixel central, lo cual no es demasiado conveniente, teniendo en cuenta que, aunque inicialmente trabajemos con imágenes ideales, el objetivo será lógicamente aplicarlo en imágenes reales, y en éstas no podemos garantizar que no haya otros contornos cerca que estén influyendo en el valor de esos pixels.

Por otro lado, obsérvese del lema 4.3 que la información que usamos para obtener los parámetros del contorno de F son las columnas $\{-1, 0, 1\}$ de la imagen G. Pero estas columnas, al haber sido generada G con una máscara de radio 1, incluyen información de las columnas $\{-2, ..., 2\}$ de la imagen original F (aunque las columnas más alejadas tengan menos peso en las expresiones). Si la convolución se hubiera hecho con una máscara de radio n, los valores de las tres columnas centrales de la imagen G se verían afectadas por las columnas $\{-(n+1), ..., (n+1)\}$ de la imagen original. Y de nuevo esto implicaría usar información demasiado alejada para poder estimar los parámetros exactos del contorno en el pixel central.

Por lo tanto, nos quedaremos sólamente con el método para imagenes suavizadas con máscaras de radio 1.

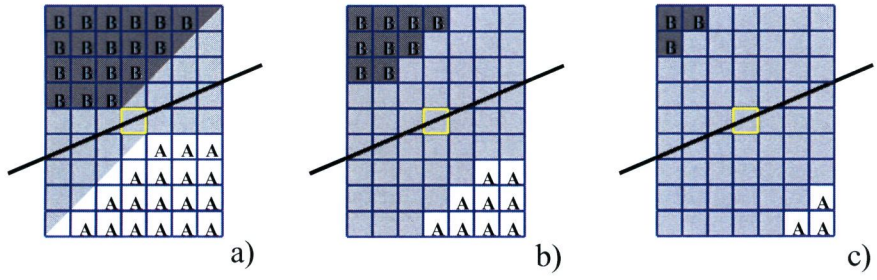


Figura 4.14: a) Imagen ideal con un contorno recto. b) Suavizado de la imagen con una máscara H_1. c) Suavizado de la imagen con una máscara H_2.

4.4 Método suavizado de segundo orden en imágenes 2D

Vamos a centrar el estudio en imágenes suavizadas con máscaras de radio 1 solamente, por los motivos antes mencionados. Para poder estimar la curvatura del contorno, simplemente cambiaremos la expresión lineal que acabamos de considerar para representar dicho contorno por una expresión cuadrática, del tipo

$$y = a + bx + cx^2$$

de la misma manera que hicimos en el capítulo previo. Por tanto, sea ahora F una imagen ideal como la de capítulos anteriores, con un contorno de segundo orden con pendiente entre 0 y 1 sobre la vertical central del pixel central, (al que llamaremos por simplicidad $F_{0,0}$), que separa dos zonas de intensidad A y B, como muestra la figura 4.15a.

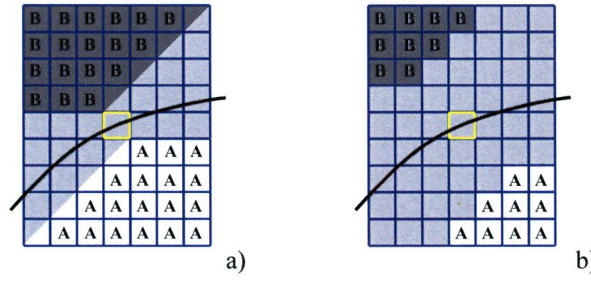


Figura 4.15: a) Imagen ideal con un contorno cuadrático. b) Suavizado de la imagen con una máscara H_1.

En principio, vamos a considerar que los pixels por los cuales puede pasar el contorno, cuyas intensidades son a priori desconocidas, (zona gris en la figura) son los mismos que para el caso lineal. Recordemos del capítulo previo que, para contornos con curvatura muy grande, podríamos estar cometiendo un error en esta consideración, aunque en la gran mayoría de los casos este hecho no va a ocurrir.

Si rehacemos todo el desarrollo previo de la sección anterior, llegamos a las mismas expresiones para las sumas S_L, S_M, S_R (ecuaciones 4.7 y 4.9) y para las sumas L_i, M_i, R_i (ecuaciones 4.8 y 4.10). La diferencia está para la expresión de las áreas P_i, Q_i, T_i, que en este caso son las siguientes:

$$
\begin{aligned}
Q_i &= \int_{(i-\frac{1}{2})h}^{(i+\frac{1}{2})h} \left(a + bx + cx^2 + \frac{7}{2}h\right) dx = ah + bih^2 + ch^3\left(\frac{1}{12} + i^2\right) + \frac{7}{2}h^2 \\
P_i &= \int_{(i-\frac{1}{2})h}^{(i+\frac{1}{2})h} \left(a + bx + cx^2 + \frac{5}{2}h\right) dx = ah + bih^2 + ch^3\left(\frac{1}{12} + i^2\right) + \frac{5}{2}h^2 \\
T_i &= \int_{(i-\frac{1}{2})h}^{(i+\frac{1}{2})h} \left(a + bx + cx^2 + \frac{3}{2}h\right) dx = ah + bih^2 + ch^3\left(\frac{1}{12} + i^2\right) + \frac{3}{2}h^2
\end{aligned} \tag{4.12}
$$

Combinándolas todas ellas, obtenemos en esta ocasión un sistema lineal de tres ecuaciones, donde las incógnitas a resolver son los coeficientes a, b, c de la parábola. Una vez resuelto, obtenemos las expresiones siguientes

$$
\begin{aligned}
a &= \frac{2S_M - 7(A+B)}{2(A-B)}h + \frac{2S_M - S_L - S_R}{24(A-B)}(1 + 24a_{01} + 48a_{11})h \\
b &= 1 + \frac{S_R - S_L}{2(A-B)} \\
c &= \frac{S_L + S_R - 2S_M}{2h(A-B)}
\end{aligned}
$$

Finalmente, con dichas expresiones podríamos entonces calcular los parámetros del contorno (posición, orientación y curvatura) evaluando la ecuación de la parábola en el punto $x = 0$. Con esto damos paso al lema final

Lema 4.4 *Sea F una imagen por cuyo pixel (i, j) pasa un contorno con pendiente entre 0 y 1, que separa dos zonas de intensidad A (debajo del contorno) y B (encima del contorno), donde se asume que en la vecindad de ese pixel, el borde se comporta como una parábola. Dicha imagen ha sido suavizada con una máscara de suavizado H_1, tal que*

$$
H_1 = \begin{pmatrix} a_{11} & a_{01} & a_{11} \\ a_{01} & a_{00} & a_{01} \\ a_{11} & a_{01} & a_{11} \end{pmatrix}
$$

dando como resultado la imagen G. A partir de los datos de la imagen suavizada G es posible obtener los parámetros del contorno de la imagen original F (posición, pendiente, magnitud del salto de intensidad y curvatura, medidos sobre la vertical central del pixel), realizando los siguientes pasos:

a) obtenemos las sumas de las columnas izquierda, central y derecha de la siguiente manera:

$$
\begin{aligned}
S_L &= \sum_{k=-2}^{4} G_{i-1,j+k} \\
S_M &= \sum_{k=-3}^{3} G_{i,j+k} \\
S_R &= \sum_{k=-4}^{2} G_{i+1,j+k}
\end{aligned}
$$

b) obtenemos también una estimación para los valores de intensidad a ambos lados del borde

$$B = \frac{G_{i-1,j-3} + G_{i-1,j-4} + G_{i,j-4}}{3}$$
$$A = \frac{G_{i,j+4} + G_{i+1,j+4} + G_{i+1,j+3}}{3}$$

c) a continuación detectamos los coeficientes de la parábola $y = a + bx + cx^2$ como sigue:

$$a = \frac{2S_M - 7(A+B)}{2(A-B)} + \frac{2S_M - S_L - S_R}{24(A-B)}(1 + 24a_{01} + 48a_{11})$$
$$b = 1 + \frac{S_R - S_L}{2(A-B)}$$
$$c = \frac{S_L + S_R - 2S_M}{2(A-B)}$$

d) finalmente, calculamos los parámetros del contorno

corte con la vertical central del pixel: $(0, a)$

vector normal: $N = \frac{A-B}{\sqrt{1+b^2}} [\ b, \quad -1\ ]$

curvatura : $K = \frac{2c}{(1+b^2)^{3/2}}$

Podemos apreciar una gran diferencia con respecto al método lineal. Para poder obtener el valor de a es preciso conocer cuáles fueron los valores empleados en la máscara de suavizado, cosa que no ocurría antes, ya que era independiente de los valores de la máscara. El resto es prácticamente similar en ambos métodos.

De hecho, llamando a',b' a los coeficientes de la recta encontrados por el método lineal, y a, b, c a los coeficientes de la parábola, encontramos la siguiente relación

$$a' = a - \frac{1 + 24a_{01} + 48a_{11}}{12}c \qquad (4.13)$$
$$b' = b$$

Es decir, la pendiente es la misma en ambos casos, y la posición es ligeramente diferente, en una cantidad proporcional al valor de c. Por ejemplo, para contornos rectos, $c = 0$, con lo cual $a = a'$. Sin embargo, para contornos con una cierta curvatura, el desplazamiento del método lineal cometerá un cierto error, mayor cuánto mayor sea c.

Al igual que se indicó en el capítulo previo, aunque el método cuadrático parezca que, al ser más general que el método lineal, debería usarse en todos los casos, esto no es así. Cuando el objetivo sea buscar rectas en una imagen, el método lineal dará mejores resultados, pues se limita a buscar exactamente lo que queremos.

Casos que producen error Al igual que se hizo en el capítulo previo, hay que tener en cuenta que contornos con curvatura muy alta pueden provocar error. Esto es debido a que no se cumple la suposición inicial que se ha hecho para el cálculo de la expresión de las áreas Q_i, P_i, T_i (ecuaciones 4.12). Recordemos que para obtener dichas expresiones se suponía que el contorno cruzaba cada una de las columnas siempre de izquierda a derecha. Si esto no es así, el método no sería exacto.

En la figura 4.16 pueden verse las 9 columnas que se están calculando sobre la imagen original F. Centrándonos en la columna más alejada, L_{-2}, podemos apreciar que para el cálculo de Q_{-2} se está usando la expresión siguiente (ver ecuación 4.12):

$$Q_{-2} = \int_{-\frac{5}{2}h}^{-\frac{3}{2}h} \left(a + bx + cx^2 + \frac{7}{2}h \right) dx$$

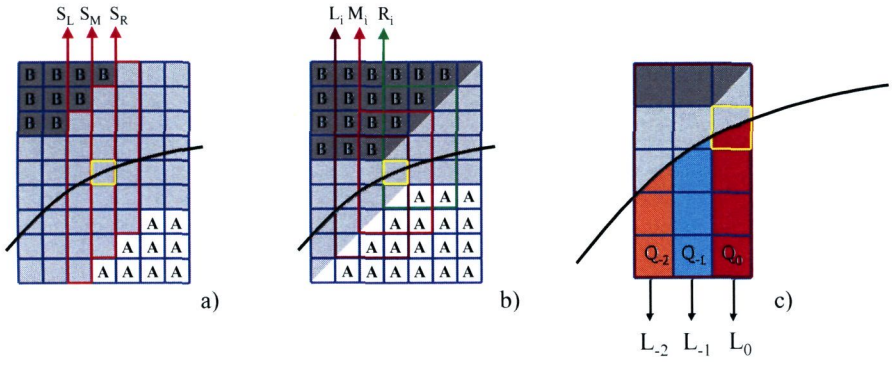


Figura 4.16: Columnas cuyas expresiones son utilizadas por nuestro método: a) Columnas de la imagen suavizada G. b) Columnas de la imagen original F. c) Detalle de las columnas L_i, donde Q_i representa el área bajo el contorno interior a cada columna.

Para que esta área Q_{-2} represente con exactitud el área bajo el contorno interior a la columna L_{-2}, es necesario que la curva no alcance el pixel $(i-2, j+4)$. Esta condición es un poco más estricta que la que teníamos en el método del capítulo anterior (que era no alcanzar el pixel $(i-1, j+3)$. De hecho, si calculásemos el radio de la menor circunferencia que, atravesando la vertical central del pixel (i, j) con pendiente menor que 1, no pueda alcanzar el pixel $(i-2, j+4)$, obtendríamos que éste es

$$R_{\min} = \frac{37}{4}\sqrt{2}h \simeq 13.08h \tag{4.14}$$

Vemos que es mayor que el del método propuesto anterior ($8.84h$). Sin embargo, a efectos numéricos, el área Q_{-2} tiene un peso menor que el resto de las columnas en las expresiones del método, ya que la columna $i-2$ de la imagen F aparece solamente en la expresión de S_L (ecuación 4.9) y además escalada por los coeficientes de la máscara de valor más pequeño (a_{01}, a_{11}). Por lo tanto, el error cometido para radios cercanos e inferiores a $R_{\min}$ será muy pequeño.

4.5 Generalización para el resto de octantes

A lo largo del capítulo nos hemos centrado solamente en los casos en donde la pendiente estaba entre 0 y 1 y $A > B$, siendo A la intensidad por debajo de la curva, y B por encima. Para generalizar al resto de octante procederemos exactamente como hicimos en el capítulo previo (ver sección 3.4). La primera generalización que hay que hacer es decidir los pixels a partir de los cuales estimaremos los valores de A y B, para cada octante en el que nos encontremos. La segunda es invertir el signo del coeficiente c en los casos en que $A < B$.

Otro cambio que debemos hacer es considerar los límites de las columnas para las sumas S_L y S_R según el octante. Por ejemplo, en la figura 4.16a puede verse que S_L está desplazada un pixel hacia abajo con respecto a S_M, y que S_R está desplazada un pixel hacia arriba. Esto lo haremos para pendientes entre 0 y 1. Pero para

pendientes entre 0 y -1, debería ser la situación simétrica, es decir, S_L un pixel más arriba que S_M, y S_R un pixel más abajo. En este último caso, la expresión 4.11 para el parámetro b sería ahora la siguiente:

$$b = -1 + \frac{S_R - S_L}{2(A-B)}$$

Con todas estas adaptaciones, ya podemos agrupar en un único lema los métodos lineal y cuadrático, para todos los casos en los que $b \in [-1, 1]$:

Lema 4.5 *Sea F una imagen por cuyo pixel (i,j) pasa un contorno que separa dos zonas de intensidad A (debajo del contorno) y B (encima del contorno). Dicha imagen ha sido suavizada con una máscara de suavizado H_1, tal que*

$$H_1 = \begin{pmatrix} a_{11} & a_{01} & a_{11} \\ a_{01} & a_{00} & a_{01} \\ a_{11} & a_{01} & a_{11} \end{pmatrix}$$

dando como resultado la imagen G, donde se cumple que

$$|G_y(i,j)| > |G_x(i,j)|$$

A partir de los datos de la imagen suavizada G podemos estimar los parámetros del contorno de la imagen original F (posición, pendiente, magnitud del salto de intensidad y curvatura, medidos sobre la vertical central del pixel), realizando los siguientes pasos:

a) obtenemos las sumas de las columnas izquierda, central y derecha de la siguiente manera:

$$\begin{aligned} S_L &= \sum_{k=-3+m}^{3+m} G_{i-1,j+k} \\ S_M &= \sum_{k=-3}^{3} G_{i,j+k} \\ S_R &= \sum_{k=-3-m}^{3-m} G_{i+1,j+k} \end{aligned}$$

donde

$$m = \begin{cases} 1 & \text{si } G_x(i,j)G_y(i,j) > 0 \\ -1 & \text{si } G_x(i,j)G_y(i,j) < 0 \end{cases}$$

b) obtenemos también una estimación para los valores de intensidad a ambos lados del borde

$$\begin{aligned} B &= \frac{G_{i-m,j-3} + G_{i-m,j-4} + G_{i,j-4}}{3} \\ A &= \frac{G_{i,j+4} + G_{i+m,j+4} + G_{i+m,j+3}}{3} \end{aligned}$$

c) a continuación detectamos los coeficientes de la ecuación $y = a + bx + cx^2$ como sigue:

$$
\begin{aligned}
c &= \begin{cases} 0 & \text{si buscamos contornos rectos} \\ \frac{S_L+S_R-2S_M}{2(A-B)} & \text{si buscamos contornos de segundo grado} \end{cases} \\
b &= m + \frac{S_R - S_L}{2(A-B)} \\
a &= \frac{2S_M - 7(A+B)}{2(A-B)} - \frac{1 + 24a_{01} + 48a_{11}}{12} c
\end{aligned}
$$

d) finalmente, calculamos los parámetros del contorno

$$
\begin{aligned}
\text{corte con la vertical central del pixel:} \quad & (0, a) \\
\text{vector normal:} \quad & N = \frac{A-B}{\sqrt{1+b^2}} \left[\; b, \quad -1 \; \right] \\
\text{curvatura} \quad : \quad & K = \frac{2cn}{(1+b^2)^{3/2}}
\end{aligned}
$$

donde

$$
n = \begin{cases} 1 & \text{si } G_y(i,j) > 0 \\ -1 & \text{si } G_y(i,j) < 0 \end{cases}
$$

Para los otro cuatro octantes, donde la pendiente es mayor que 1 en valor absoluto, se sigue un esquema simétrico, intercambiando las variables x e y, y trabajando por filas en lugar de por columnas. Ahora el objetivo es encontrar los coeficientes de la ecuación $x = a+by+cy^2$ que representa el contorno. A continuación se muestra el lema para estos cuatro octantes.

Lema 4.6 *Sea F una imagen por cuyo pixel (i,j) pasa un contorno que separa dos zonas de intensidad A (debajo del contorno) y B (encima del contorno). Dicha imagen ha sido suavizada con una máscara de suavizado H_1, tal que*

$$
H_1 = \begin{pmatrix} a_{11} & a_{01} & a_{11} \\ a_{01} & a_{00} & a_{01} \\ a_{11} & a_{01} & a_{11} \end{pmatrix}
$$

dando como resultado la imagen G, donde se cumple que

$$
|G_y(i,j)| < |G_x(i,j)|
$$

A partir de los datos de la imagen suavizada G podemos estimar los parámetros del contorno de la imagen original F (posición, pendiente, magnitud del salto de intensidad y curvatura, medidos sobre la horizontal central del pixel), realizando los siguientes pasos:

a) obtenemos las sumas de las filas superior, central e inferior de la siguiente manera:

$$
\begin{aligned}
S_L &= \sum_{k=-3+m}^{3+m} G_{i+k,j-1} \\
S_M &= \sum_{k=-3}^{3} G_{i+k,j} \\
S_R &= \sum_{k=-3-m}^{3-m} G_{i+k,j+1}
\end{aligned}
$$

donde

$$m = \begin{cases} 1 & \text{si } G_x(i,j)G_y(i,j) > 0 \\ -1 & \text{si } G_x(i,j)G_y(i,j) < 0 \end{cases}$$

b) obtenemos también una estimación para los valores de intensidad a ambos lados del borde

$$\begin{aligned} B &= \frac{G_{i-3,j-m} + G_{i-4,j-m} + G_{i-4,j}}{3} \\ A &= \frac{G_{i+4,j} + G_{i+4,j+m} + G_{i+3,j+m}}{3} \end{aligned}$$

c) a continuación detectamos los coeficientes de la ecuación $x = a + by + cy^2$ como sigue:

$$\begin{aligned} c &= \begin{cases} 0 & \text{si buscamos contornos rectos} \\ \frac{S_L + S_R - 2S_M}{2(A-B)} & \text{si buscamos contornos de segundo grado} \end{cases} \\ b &= m + \frac{S_R - S_L}{2(A-B)} \\ a &= \frac{2S_M - 7(A+B)}{2(A-B)} - \frac{1 + 24a_{01} + 48a_{11}}{12} c \end{aligned}$$

d) finalmente, calculamos los parámetros del contorno

corte con la vertical central del pixel: $(a, 0)$

vector normal: $N = \frac{A-B}{\sqrt{1+b^2}} \left[\; 1, \quad -b \; \right]$

curvatura : $K = \frac{2cn}{(1+b^2)^{3/2}}$

donde

$$n = \begin{cases} 1 & \text{si } G_x(i,j) > 0 \\ -1 & \text{si } G_x(i,j) < 0 \end{cases}$$

4.6 Algoritmo para la localización de contornos en imágenes suavizadas

Con los lemas anteriores ya estamos en disposición de desarrollar el algoritmo para tratar de detectar todos los parámetros del contorno (posición, pendiente, salto de intensidad y curvatura) en una imagen suavizada. Al igual que en los otros métodos, lo primero es detectar en la imagen qué pixels podemos etiquetar como borde, a partir de las imágenes de parciales G_x y G_y, generadas mediante una convolución con sendas máscaras H_x y H_y. Estamos interesados en aquellos pixels que cumplan

$$G_x^2(i,j) + G_y^2(i,j) > \delta^2$$

donde δ es un umbral que habrá que prefijar. Ya hemos indicado que esta condición no es en realidad relevante para la teoría que hemos desarrollado, pero en la práctica es necesaria para evitar que se detecten bordes en pixels donde apenas hay un cambio de intensidad leve. La segunda condición sí es más importante: debe cumplirse que

$$|G_y(i,j-2)| < |G_y(i,j-1)| < |G_y(i,j)| > |G_y(i,j+1)| > |G_y(i,j+2)|$$

en el caso en que $|G_y(i,j)| > |G_x(i,j)|$, o bien que

$$|G_x(i-2,j)| < |G_x(i-1,j)| < |G_x(i,j)| > |G_x(i+1,j)| > |G_x(i+2,j)|$$

en el caso en que $|G_y(i,j)| < |G_x(i,j)|$. Esta última condición era para evitar puntos aislados que no correspondan a un perfil como el que se estudió en la sección 2.1.1.

Una vez tenemos los pixels identificados, aplicaremos alrededor de cada uno el esquema desarrollado en la sección anterior para obtener los parámetros estimados para el contorno. Esto significa que, para cada pixel borde, el método usará los valores de un entorno centrado en dicho pixel, cuyas dimensiones serán 9×3 (para el caso en el que $|G_y(i,j)| > |G_x(i,j)|$) ó 3×9 en caso contrario.

El pseudo-código de nuestro algoritmo podría ser el siguiente:

Funcion ContornosSuavizados (delta)

```
Calcular Gx, Gy
Para todos los pixels (i,j) de la imagen
   Si Gx(i,j)*Gx(i,j) + Gy(i,j)*Gy(i,j) < delta * delta Continuar
   Si |Gy(i,j)| > |Gx(i,j)| Entonces
      Si |Gy(i,j)| no es maximo en su columna Continuar
      Calcular Curvatura, Vector Normal y Posicion segun Lema 4.5
         Si No
      Si |Fx(i,j)| no es maximo en su fila Continuar
      Calcular Curvatura, Vector Normal y Posicion segun Lema 4.6
   Fin Si
Fin Para
```

4.7 Ejemplos

A continuación vamos a probar nuestro nuevo método en diversos tipos de imágenes.

4.7.1 Contorno ideal recto

Tomemos como primer ejemplo una imagen sintética de un contorno de 30 grados, con un salto de intensidad entre ambos lados del contorno de 100 unidades, y que ha sido suavizada con una máscara de radio 1. A esta imagen suavizada le aplicamos tres métodos: el método clásico visto en el capítulo 1, el método lineal propuesto en el capítulo previo (MPG1) y el método lineal propuesto actual (MPISG1). Los resultados se muestran en las tres primeras filas de la tabla 4.2.

Podemos ver como nuestro método previo sigue funcionando mejor que el método tradicional en la imagen suavizada, aunque comete algunos errores debido a la difuminación del contorno producida por el suavizado. Sin embargo, el método propuesto actual obtiene con exactitud los parámetros de la recta en cada pixel borde. En la fila superior de la figura 4.17 podemos ver, de izquierda a derecha, la imagen original, la imagen suavizada, el resultado de aplicar nuestro método previo a la imagen suavizada, y el resultado de aplicar el método propuesto actual a dicha imagen.

Añadamos ahora algo de ruido a la imagen original para ver el comportamiento de los métodos. Utilizaremos como modelo de ruido un ruido blanco gaussiano aditivo de media cero y desviación típica σ. Esto significa que

		Err. módulo		Err. dirección		Err. posición	
Imagen	**Método**	Media	Max.	Media	Max.	Media	Max.
sin ruido	Tradicional	26.9	30.1	1.26	1.61	*NC*	*NC*
	MPG1	2.09	3.56	0.40	0.50	12.8	26.6
	MPISG1	0.00	0.00	0.00	0.01	0.00	0.00
ruido 5	Tradicional	26.7	31.6	1.54	4.91	*NC*	*NC*
	MPG1	2.00	5.96	1.34	4.83	13.4	36.7
	MPISG1	1.45	4.74	1.47	4.53	7.48	21.8
ruido 10	Tradicional	25.9	39.8	2.29	7.13	*NC*	*NC*
	MPG1	3.95	11.6	2.28	7.20	14.6	42.6
	MPISG1	3.11	9.70	2.75	8.44	13.3	39.0

Tabla 4.2: Resultados obtenidos por cada método al aplicarse sobre el suavizado de una imagen con contorno recto con ruido 0 (parte superior de la tabla), 5 (parte central) y 10 (parte inferior). MPG1: método propuesto original de grado 1. MPISG1: método propuesto para imágenes suavizadas de grado 1. NC: no lo calcula

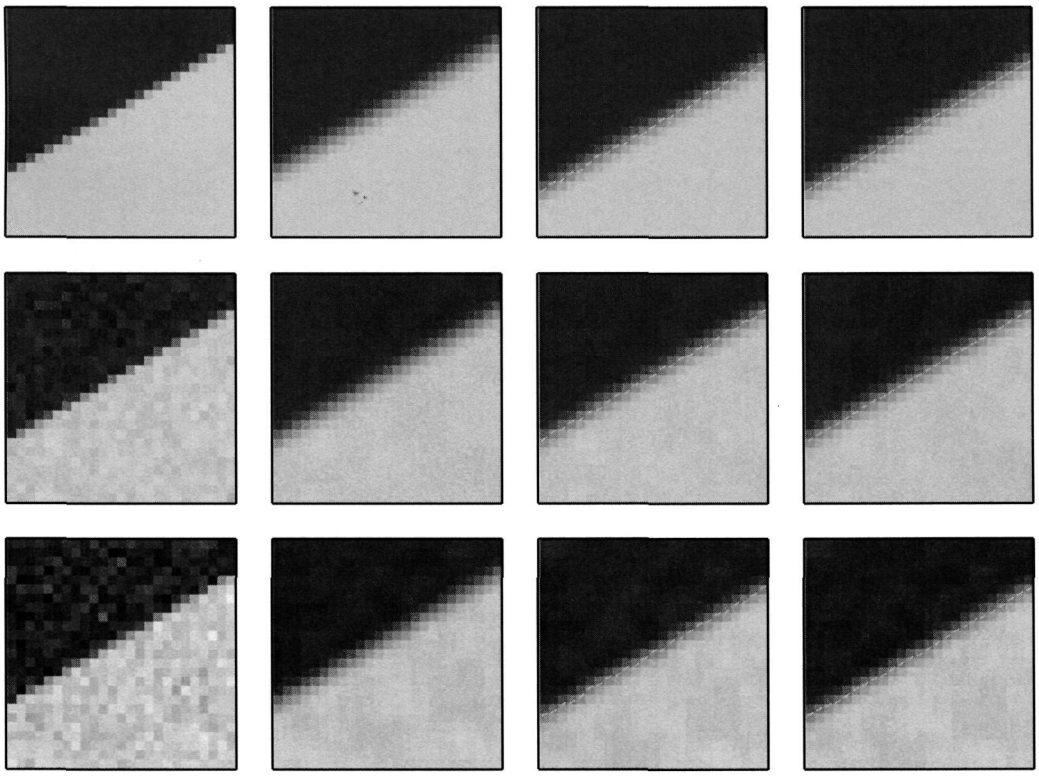


Figura 4.17: De izquierda a derecha: imagen ideal con contorno recto sin ruido (fila superior), con ruido 5 (fila central) y con ruido 10 (fila inferior). Imagen suavizada. Método propuesto previo aplicado a la imagen suavizada. Método propuesto actual aplicado a la imagen suavizada

		Err. módulo		Err. dirección		Err. posición		Radio de curvatura		
Imagen	**Método**	Media	Max.	Media	Max.	Media	Max.	Media	Mín.	Máx.
sin ruido	Tradicional									
	MPG2	7.18	11.1	1.17	1.63	8.42	18.9	−15.3	−21.3	−13.2
	MPISG2	0.00	0.00	0.24	0.71	0.07	0.29	−14.8	−14.9	−14.6
ruido 5	Tradicional									
	MPG2	7.67	13.2	1.74	5.89	8.60	25.6	−15.6	–	–
	MPISG2	1.26	3.93	1.65	5.87	8.80	33.6	−14.3	–	–
ruido 10	Tradicional									
	MPG2	7.98	18.2	2.99	13.4	12.3	46.5	−14.7	–	–
	MPISG2	3.45	8.78	3.91	14.4	16.4	49.4	−14.2	–	–

Tabla 4.3: Resultados obtenidos por cada método al aplicarse sobre el suavizado de una imagen con contorno circular con ruido 0 (parte superior de la tabla), 5 (parte central) y 10 (parte inferior). MPG2: método propuesto original de grado 2. MPISG2: método propuesto para imágenes suavizadas de grado 2. NC: no lo calcula

a cada pixel se le sumará un valor aleatorio tomado de una distribución normal, de media nula y desviación σ. De esta forma, el parámetro σ mide en cierta manera la magnitud del ruido que se le añade a la señal.

En las filas restantes de la tabla 4.2 se muestran los resultados de aplicar los tres métodos, primero a la imagen original a la que se le ha añadido un ruido de magnitud 5, y luego con magnitud 10. Ambas fueron posteriormente suavizadas también con una máscara de radio 1, antes de aplicarles los métodos. En las filas central e inferior de la figura 4.17 se muestra (de izquierda a derecha) la imagen con ruido, la suavizada, y el resultado de aplicar los métodos propuesto anterior y actual respectivamente a esta última imagen.

Puede verse cómo cuando el ruido no es demasiado grande, el método actual produce valores más suavizados que el método previo. Sin embargo, cuando el ruido empieza a ser excesivo, ya el error en ambos métodos empieza a ser muy alto. En el capítulo siguiente abordaremos el problema de localizar los contornos cuando el ruido sea demasiado fuerte.

4.7.2 Contorno ideal circular

Hagamos lo mismo con una imagen ideal en la que aparece un círculo de radio 15 pixels, con un salto de intensidad de 100 unidades. Primero suavizamos con una máscara de radio 1 y luego aplicamos los tres métodos anteriores. El resultado se observa en las tres primeras filas de la tabla 4.3. En la fila superior de la figura 4.18 podemos ver, de izquierda a derecha, la imagen original, la imagen suavizada, el resultado de aplicar nuestro método cuadrático previo (MPG2) a la imagen suavizada, y el resultado de aplicar el método cuadrático propuesto actual (MPISG2) a dicha imagen.

De nuevo vemos que nuestro método previo sigue funcionando mejor que el método tradicional en la imagen suavizada, aunque comete algunos errores debido a la difuminación del contorno producida por el suavizado. Sin embargo, el método propuesto actual obtiene con más exactitud los parámetros de la circunferencia en cada pixel borde. Sin embargo, siendo rigurosos, hay que notar que existen pequeños errores en la dirección y en la posición. Esto es debido a que, al tomar más pixels en consideración de la imagen original que con el método propuesto previo[1], las diferencias entre un arco de parábola y un arco de circunferencia son mayores. En el capítulo siguiente corregiremos también este problema.

En las filas restantes de la tabla 4.3 se muestran los resultados de aplicar los tres métodos, primero a la imagen original a la que se le ha añadido un ruido de magnitud 5, y luego con magnitud 10. Ambas fueron posteriormente suavizadas también con una máscara de radio 1, antes de aplicarles los métodos. En las filas

[1] Aunque sólo tomamos 3 columnas en la imagen suavizada, indirectamente estamos usando 5 columnas de la imagen original.

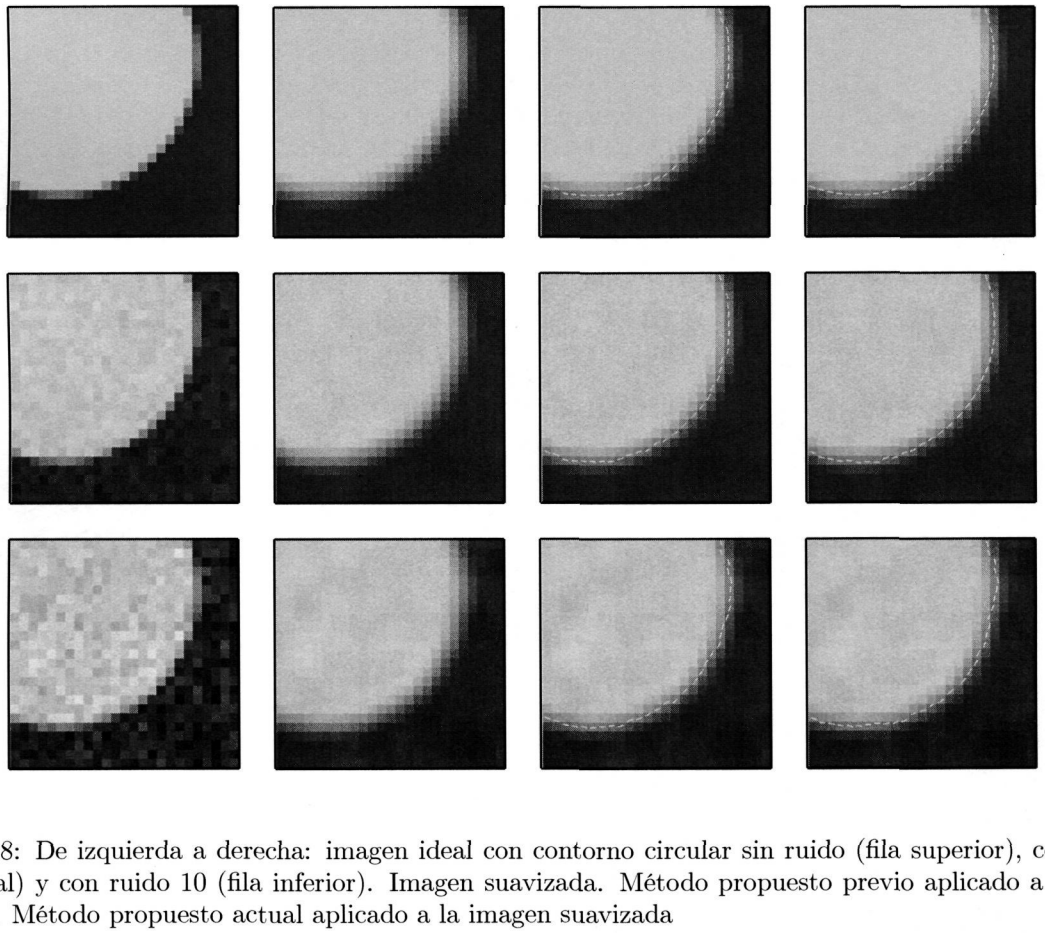


Figura 4.18: De izquierda a derecha: imagen ideal con contorno circular sin ruido (fila superior), con ruido 5 (fila central) y con ruido 10 (fila inferior). Imagen suavizada. Método propuesto previo aplicado a la imagen suavizada. Método propuesto actual aplicado a la imagen suavizada

central e inferior de la figura 4.18 se muestra (de izquierda a derecha) la imagen con ruido, la suavizada, y el resultado de aplicar los métodos propuesto anterior y actual respectivamente a esta última imagen. Volvemos a ver que cuando el ruido no es demasiado grande, el método actual produce valores más homogéneos que el método previo, pero a medida que el ruido crece el error en ambos métodos se dispara.

4.7.3 Contornos reales

Probemos ahora con las imágenes reales utilizadas en capítulos anteriores. Comenzando con la imagen de la diana, hemos aplicado un suavizado y posteriormente hemos aplicado el último método propuesto. Si rehiciésemos las tablas 2.4 y 2.6, veríamos que los resultados numéricos son similares que con los métodos anteriores, incluso ligeramente peores.

Esto podría parecer una contradicción. Sin embargo, hay que tener en cuenta que el proceso de suavizado ha mezclado la información de la señal original con el ruido producido en la adquisición, con lo cual la recuperación de la señal original exacta es muy difícil. Sin embargo, visualmente los valores obtenidos con el nuevo método propuesto después de suavizar son bastante más homogéneos entre sí, produciendo una línea de contorno más suave entre pixels contiguos que con los métodos anteriores.

Esto puede verse en la figura 4.19 donde se muestra, de izquierda a derecha, nuestro método previo aplicado a la imagen original y a la suavizada, y por último nuestro método final aplicado a la imagen suavizada. Es en esta última columna donde se obtienen los mejores resultados.

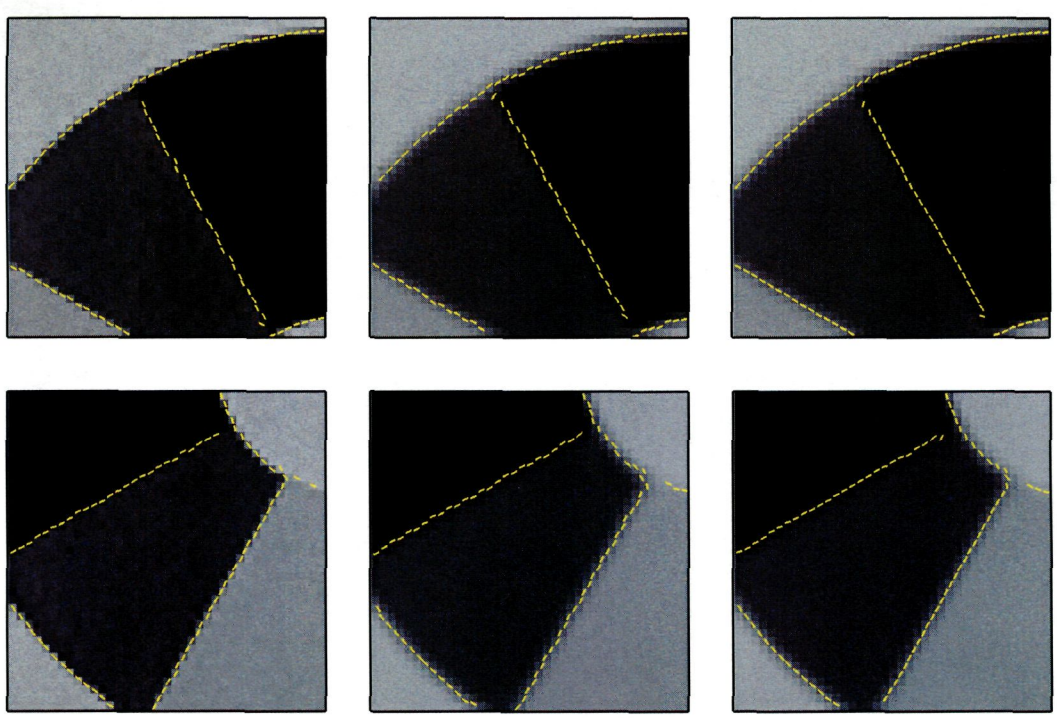


Figura 4.19: De izquierda a derecha: Método propuesto inicial sobre la imagen original. Método propuesto inicial sobre la imagen suavizada. Método propuesto último sobre la imagen suavizada.

Con la imagen del Popeye ocurre exactamente lo mismo. El resultado ahora es bastante más homógeneo que con los métodos anteriores, como puede verse en la columna derecha de la figura 4.20.

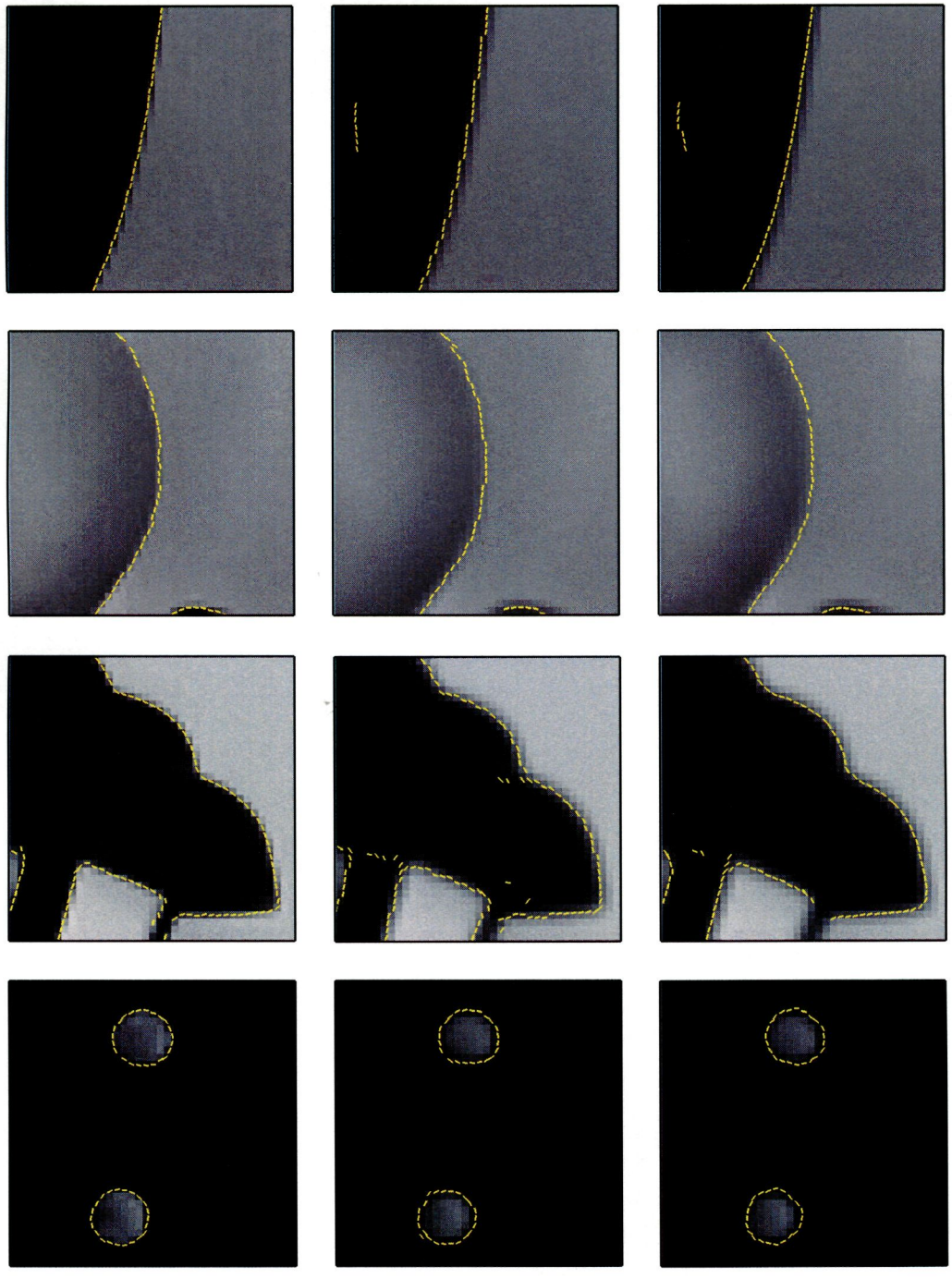


Figura 4.20: De izquierda a derecha: Método propuesto inicial sobre la imagen original. Método propuesto inicial sobre la imagen suavizada. Método propuesto último sobre la imagen suavizada.

4.8 Resultado de aplicar el método a contornos que estén desenfocados

En el modelo de adquisición que propusimos en la sección 1.2 usábamos la suposición de que cada punto del espacio tridimensional que se quería fotografiar se proyectaba sobre un único punto del plano de proyección. Es lo que se suele llamar una **cámara de tipo pin-hole.** En este tipo de cámaras, todos los objetos aparecen enfocados, es decir, todos los contornos de la imagen aparecen igual de nítidos, independientemente de la lejanía del objeto con respecto a la posición de la cámara.

Para generalizar un poco más el modelo, con la intención de acercarmos más a la realidad, podemos usar el **modelo de lente delgada**, en el cual la óptica es modelada como una única lente que concentra en un punto todos los rayos de luz procedentes de un determinado punto del espacio [JAH95]. En la figura 4.21 se muestra el proceso de formación de la imagen con este modelo.

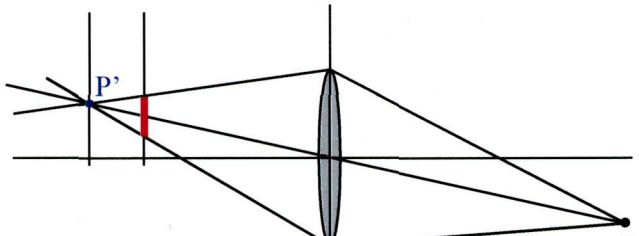


Figura 4.21: Proceso de formación de la imagen con el modelo de lente delgada. El punto P está enfocado sólamente cuando el plano imagen está a distancia d azul.

En dicha figura podemos ver cómo los rayos de luz procedentes de un punto P del espacio son desviados por la lente para concentrarse en un solo punto P' en el interior de la cámara. Cuando la distancia d entre la lente y el plano imagen coincide con la distancia a la que se encuentra P', el punto P es proyectado en un único punto sobre el plano imagen, y se dice que está enfocado. Pero si la distancia al plano imagen es diferente, el punto no se proyecta en un único punto P', sino en un área llamada **círculo de confusión**, y es lo que produce el desenfoque del punto.

Con este nuevo modelo, la imagen de un contorno puede ser bastante diferente a la que estábamos usando. Cojamos un contorno vertical, como el de la figura 4.22, que separa dos niveles de intensidad, A y B a ambos lados. Con el modelo pin-hole, cada punto del espacio es proyectado en un único punto del plano imagen. Se produce por tanto una discontinuidad en la señal $f(x,y)$ recibida en el sensor. Posteriormente la discretización produce una señal $F(x_i, y_j)$ con una única columna con intensidad intermedia entre A y B.

Por el contrario, con el modelo de lente delgada, cada punto del espacio se proyecta en una zona circular sobre el plano imagen, con lo cual cada punto (x,y) de la imagen recibe intensidad de una zona equivalente en el espacio. Esto provoca una difuminación en la señal, produciendo una señal discretizada con varias columnas de intensidad intermedia entre A y B. En el anexo B se ha obtenido la expresión para $f(x)$ según este modelo, tomando un caso unidimensional. Esta expresión es la siguiente

$$f(x) = \begin{cases} B & x < t - R \\ \frac{A+B}{2} - \frac{A-B}{\pi}\left(\arcsin u + u\sqrt{1-u^2}\right) & t - R < x < t + R \\ A & x > t + R \end{cases}$$

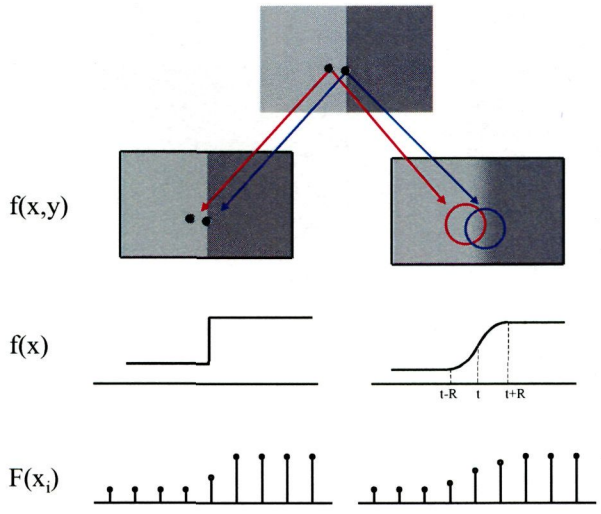


Figura 4.22: Formación de la imagen con el modelo de pin-hole (columna izquierda) y con el modelo de lente delgada (columna derecha). De arriba a abajo: Contorno vertical en el espacio. Intensidad que llega a la cámara, $f(x, y)$. La misma vista en una sola fila de la imagen. Imagen discretizada.

donde

$$u = \frac{t - x}{R}$$

donde t representa la posición exacta del contorno y R indica el radio del círculo de confusión.

Con este modelo ya podemos dar explicación a una imagen en donde exista un cierto desenfoque en algunos de los contornos. En la figura 4.23 se muestra una fotografía en donde puede apreciarse como la zona de los contornos aparece desenfocada.

Ahora un contorno ya no es una discontinuidad entre dos niveles de intensidad, A y B, sino una franja paralela al contorno, de $2R$ unidades de grosor, en donde la intensidad cambia progresivamente de A a B. Si el radio del círculo de confusión, R, es relativamente pequeño, **los métodos propuestos en este trabajo siguen siendo válidos**, siempre y cuando ocurra que la totalidad de la franja cruce de izquierda a derecha (en los bordes de pendiente menor que 1) la ventana sobre la que centraba cada pixel para obtener los parámetros del contorno, como se muestra en la figura 4.24.

La explicación es sencilla. Como todos los métodos parten de la idea de que la suma de los pixels de una columna es proporcional al área de la columna bajo la recta, esto se sigue cumpliendo cuando la franja cruza la columna de izquierda a derecha en su totalidad. Es decir, el método funcionaría en la figura b) pero no en la c), ya que

$$S_a = S_b \neq S_c$$

En la figura 4.25 se muestran varias zonas ampliadas de la imagen de la figura 4.23 donde se han utilizado los dos métodos propuestos (original y gaussiano). También se ha suavizado la imagen con una máscara $3x3$ y se han vuelto a aplicar ambos métodos. Vemos que de los cuatro casos diferentes, el método gaussiano sobre la imagen suavizada es el que mejor resultado visual da.

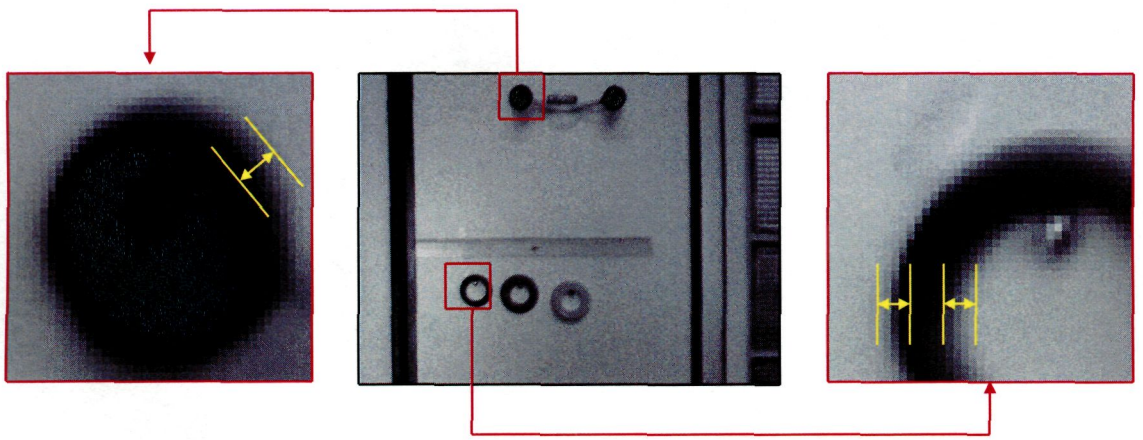


Figura 4.23: Contornos desenfocados

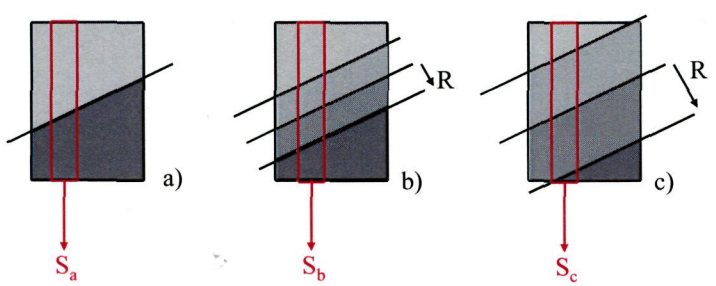


Figura 4.24: a) Contorno con el modelo pin-hole. b y c) Contornos con el modelo de lente delgada. $S_a = S_b \neq S_c$

4.9 Conclusiones

Se ha desarrollado un esquema capaz de encontrar en imágenes ideales sintéticas, previamente convolucionadas con una máscara de suavizado de radio 1, la orientación, curvatura y localización sub-pixel de un contorno de segundo grado de forma exacta, así como el salto de intensidad producido a ambos lados del borde. Posteriormente, y a partir de dicho esquema, hemos desarrollado un método para localizar los contornos en imágenes reales, cuyo funcionamiento es bastante bueno aunque la imagen tenga algo de ruido, e incluso aunque los contornos estén ligeramente desenfocados, como se ha visto en las pruebas realizadas.

Sin embargo, cuando el ruido es demasiado alto (mayor que el empleado en la figura 4.18), el obtener los contornos con precisión se hace más delicado, ya que un suavizado más fuerte provocaría una difuminación demasiado alta. En el próximo capítulo se desarrollará un esquema iterativo que irá realzando los contornos y eliminando ruido de forma simultánea.

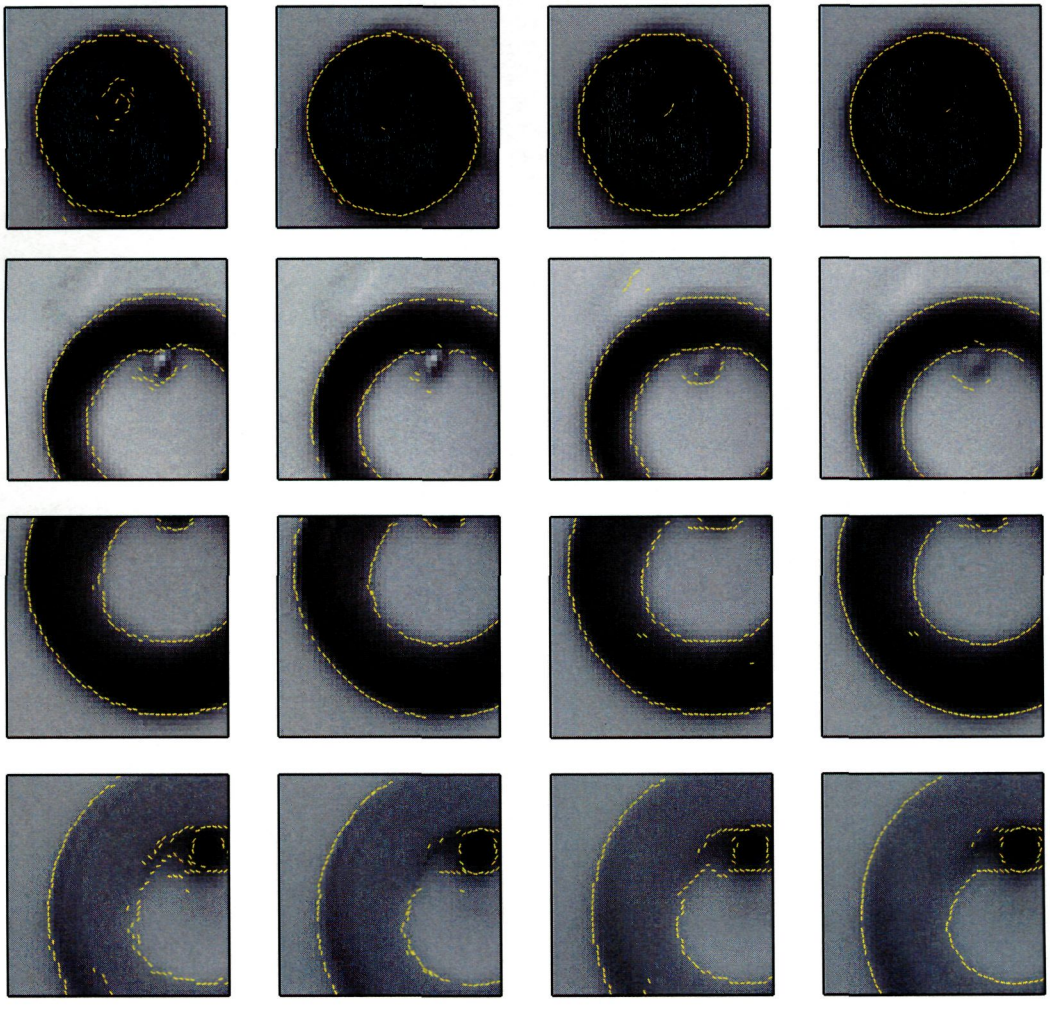


Figura 4.25: De izquierda a derecha: Métodos propuestos inicial y final sobre la imagen original. Ídem sobre la imagen suavizada.

Capítulo 5

Restauración básica de imágenes

- Aún suavizando la imagen, si el ruido es muy grande, la precisión es bastante mala.
- Pues habría que tratar de mejorar la calidad de la imagen entonces, al estilo de los métodos de restauración.
- Pero necesitaríamos uno que se ajustara a nuestras hipótesis.....

FEBRERO 2002

Ya hemos visto que la adquisición, digitalización y manipulación de imágenes introduce un cierto número de perturbaciones en la imagen original que dificultan en muchas ocasiones su correcta interpretación. Es lo que se conoce como **ruido** en la imagen. Cuando la magnitud de la perturbación no es excesiva, el método desarrollado en el capítulo previo es una buena herramienta para solventar el problema. Pero cuando el ruido es grande, hay que buscar otros métodos.

En estos casos, lo usual es definir un proceso iterativo que vaya modificando gradualmente la imagen, tratando de eliminar ruido en cada iteración, de forma que al final del proceso obtengamos una imagen más limpia que la original. Es lo que se conoce como realzado o **restauración de imágenes**. Finalmente, los parámetros del contorno se evalúan sobre la imagen restaurada.

En el capítulo anterior vimos que convolucionar con una gaussiana disminuía el ruido presente en la imagen. Por tanto, un proceso iterativo sencillo que podría definirse sería ir aplicando gaussianas sucesivamente a la imagen, hasta que el ruido desaparezca por completo. El problema de este esquema es que, junto con el ruido, se irían perdiendo también todos los contornos, bien porque se irían difuminando, bien porque se fusionarían con otros contornos cercanos. En la figura 5.1 puede verse la evolución que tendría un proceso iterativo de este tipo sobre una imagen. Si continuásemos este proceso indefinidamente, obtendríamos una imagen con una única intensidad, igual a la intensidad media de toda la imagen original.

En el presente capítulo comenzaremos viendo algunos de los procesos que suelen usarse para restaurar la imagen, y a continuación expondremos dos métodos nuevos (uno básico y otro optimizado) a partir del esquema desarrollado en el capítulo anterior, primero de grado uno y luego de grado dos, y luego someteremos ambos esquemas a pruebas con imágenes sintéticas y reales.

Figura 5.1: Proceso iterativo de convolucionar la imagen con máscaras de suavizado de radio 1. Número de iteraciones (de izquierda a derecha): 0, 20, 100 y 1000.

5.1 Técnicas tradicionales de restauración de imágenes

Una de las maneras más extendidas de abordar el problema de la restauración de imágenes consiste en buscar la imagen restaurada como el mínimo de un cierto funcional de energía. Es lo que se conoce como **formulación variacional** del problema [ROM94, SAP01, AUB02], cuyos principios básicos trataremos de describir a continuación.

5.1.1 Formulación variacional del problema

Comencemos primero por el caso continuo. Vamos a denotar por f_0 la imagen inicial con ruido de la que partimos, y por f la imagen restaurada. En el funcional de energía que vamos a considerar aparecen dos tipos de términos: un primer término, denominado término de atracción, fuerza a que f se parezca a f_0, y es de la forma

$$\int_\Omega (f - f_0)^2$$

donde Ω es el dominio donde está definida la imagen. El segundo término, denominado término de regularización, fuerza a que f no posea fuertes discontinuidades. Existen diferentes formas de expresar este hecho. La más sencilla consiste en considerar el cuadrado del módulo del vector gradiente de f, es decir, el funcional siguiente:

$$\int_\Omega \|\nabla f\|^2$$

La energía que se va a minimizar es un balance entre estos dos términos, es decir:

$$E_t(f) = \int_\Omega (f - f_0)^2 + t \int_\Omega \|\nabla f\|^2$$

donde $t \geq 0$ es un parámetro que determina el balance entre los dos términos. Vamos a denotar por f_t al mínimo del funcional $E_t(f)$, es decir $E_t(f_t) \leq E_t(f)$ para cualquier imagen f. El parámetro t representa un factor de escala. Así, cuando $t = 0$, el mínimo de $E_t(f)$ es obviamente $f_t = f_0$. Cuanto mayor sea t, más regularizada estará la imagen f_t. De hecho, cuando t tiende hacia infinito, f_t tiende hacia un valor constante dado por el valor medio de f_0 en el dominio Ω, es decir

$$\lim_{t \to \infty} f_t = \frac{1}{|\Omega|} \int_\Omega f_0$$

Ecuaciones de Euler-Lagrange Buscar exhaustivamente el mínimo del funcional $E_t(f)$ evaluándolo para todas las funciones posibles f es una tarea muy costosa computacionalmente, pues el número de funciones posibles f es gigantesco. Afortunadamente existe una técnica de búsqueda de mínimos mucho más eficiente, basada en las denominadas ecuaciones diferenciales de Euler-Lagrange asociadas a la energía $E_t(f)$, que explicaremos a continuación.

Dada una imagen cualquiera f, vamos a considerar la función

$$\phi(\lambda) = E_t(f_t + \lambda f) = \int_\Omega (f_t + \lambda f - f_0)^2 + t \int_\Omega \|\nabla f_t + \lambda \nabla f\|^2$$

Puesto que f_t es mínimo de $E_t(f)$, la función $\phi(\lambda)$ tendrá un mínimo en $\lambda = 0$, y por tanto su parcial con respecto a λ será nula en $\lambda = 0$, esto es

$$\frac{\partial \phi}{\partial \lambda}(0) = 2 \int_\Omega (f_t - f_0) f + 2t \int_\Omega \nabla f_t \nabla f = 0$$

Aplicando el teorema de la divergencia se puede transformar la expresión anterior en la siguiente:

$$\frac{\partial \phi}{\partial \lambda}(0) = 2 \int_\Omega ((f_t - f_0) - t div\,(\nabla f_t))\, f = 0$$

Por tanto, como esta igualdad se verifica para cualquier función f, debe cumplirse que

$$(f_t - f_0) - t div\,(\nabla f_t) = 0 \tag{5.1}$$

que es una ecuación en derivadas parciales denominada ecuación de Euler-Lagrange asociada al funcional $E_t(f)$. La solución de dicha ecuación nos dará la función restaurada f_t que buscamos.

Análisis numérico Para poder implementar la técnica anterior sobre una imagen digitalizada f y encontrar la imagen restaurada f_t es preciso discretizar la ecuación de Euler-Lagrange anterior y calcular su solución f_t en cada pixel. Para ello hay que discretizar el operador laplaciano

$$\Delta f_t = div\,(\nabla f_t)$$

Existen muchas formas de hacerlo. Una forma sencilla consiste en convolucionar la imagen con la máscara

$$\frac{1}{3h^2} \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Es decir, para cada pixel (i, j), su laplaciano es igual a

$$\Delta F_t(i,j) = \frac{1}{3h^2} \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix} * F_t(i,j)$$

donde h representa el tamaño del pixel. Por lo tanto, la ecuación de Euler-Lagrange evaluada en cada pixel (i, j) nos da la relación

$$F_t(i,j) - F_0(i,j) - t\Delta F_t(i,j) = 0$$

lo que genera un sistema lineal de ecuaciones donde las incógnitas son $F_t(i, j)$. Para resolver este sistema, el método más sencillo es Gauss-Seidel, donde se aproxima iterativamente la solución inicializando $F_t(i, j) = F_0(i, j)$, y haciendo iteraciones de la forma

$$F_t(i,j) = F_0(i,j) + t\Delta F_t(i,j) \tag{5.2}$$

hasta la convergencia de $F_t(i, j)$.

5.1.2 Evitar la difusión de los contornos

El caso anterior de funcional elegido $E_t(f)$ es el más sencillo, ya que tiene la ventaja de que el operador diferencial que sale en la ecuación de Euler-Lagrange es el laplaciano, y por tanto lineal. Pero tiene un inconveniente importante, y es que la restauración realizada no respeta las discontinuidades de los bordes de los objetos, produciendo un difuminado desagradable visualmente.

De hecho, es fácil demostrar cómo el esquema 5.2 es equivalente a realizar el siguiente proceso iterativo:

$$F_{n+1}(i,j) = \frac{1}{1+3t} F_0(i,j) + \frac{3t}{1+3t} \overline{F_n(i,j)}$$

donde $F_n(i,j)$ representa el pixel (i,j) en la imagen de la iteración n, t es el parámetro de escala, y $\overline{F_n(i,j)}$ es el valor promedio de una vecindad 3×3 centrada en el pixel (i,j). Es decir, en cada iteración, cada pixel se actualiza a un valor intermedio entre el valor inicial y el valor promedio de su vecindad. Cuanto mayor sea el valor de t, mayor será la difusión en todos los pixels, y el proceso se asemejará a un simple suavizado. Este proceso se conoce como **difusión isotrópica.**

Para evitar está difusión en toda la imagen, se utiliza un término de regularización que tienda a uniformizar la imagen en las zonas homogéneas, y deje intactos los bordes bien definidos. Eso se consigue utilizando una energía del tipo

$$E_t(f) = \int_\Omega (f - f_0)^2 + t \int_\Omega \Phi\left(\|\nabla f\|\right) \tag{5.3}$$

y adaptando convenientemente la función Φ. Este proceso se conoce como **difusión anisotrópica**, ya que la difusión no es igual en todos los pixels [WEI98].

Utilizando el mismo razonamiento que en la sección anterior podemos deducir que la ecuación de Euler-Lagrange asociada al funcional es

$$2\left(f_t - f_0\right) - tdiv\left(\Phi'\left(\|\nabla f\|\right) \frac{\nabla f_t}{\|\nabla f\|}\right) = 0 \tag{5.4}$$

Para cada función Φ obtendremos un esquema discretizado diferente, el cual dará lugar a un sistema de ecuaciones no lineales que habrá que resolver con algún método numérico adecuado.

Formulación de Perona-Malik La primera formulación elegante de difusión anisotrópica fue introducida por Perona y Malik [PER90], y desde entonces ha habido una cantidad considerable de investigación tanto teórica como práctica de este método de restauración: propiedades matemáticas de este tipo de difusión [CAT92, ALV93, YOU96], implementaciones eficientes [COT93, ACT94], funciones de difusión para aplicaciones específicas [WHI93, STE94, WEI94a], relaciones entre la difusión anisotrópica y otros métodos de análisis de imágenes [GEI91, SNY95], etc.

Dos de las funciones Φ propuestas por Perona y Malik son las siguientes:

$\Phi(s)$	$\Phi'(s)/s$
$k^2/2\left(1 - e^{-s^2/k^2}\right)$	e^{-s^2/k^2}
$k^2/2 \log\left(1 + s^2/k^2\right)$	$k^2/\left(k^2 + s^2\right)$

donde k es un parámetro que indica a partir de qué nivel de gradiente consideramos que una zona es homogénea o pertenece al borde de un objeto. Como puede apreciarse en la ecuación 5.4, la expresión $\Phi'(s)/s$ es la que tiene interés, ya que es la que está pesando al vector gradiente dentro de la divergencia. En la ecuación original 5.1, dicho peso es 1, lo cual implica que $\Phi(s) = s^2/2$. En la figura 5.2 pueden verse las gráficas de las dos funciones propuestas. Ambas son decrecientes, estando cercanas a uno cuando el gradiente es bajo (zonas homogéneas) y anulándose para valores más altos del gradiente (bordes).

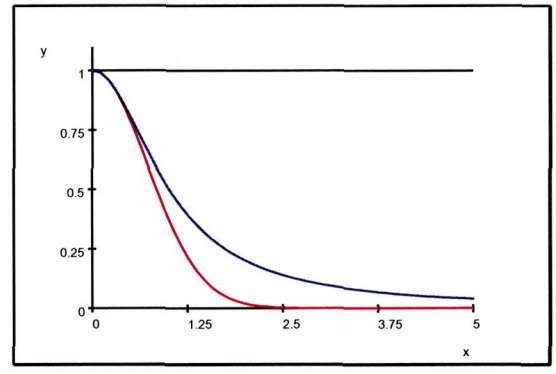


Figura 5.2: Funciones propuestas por Perona-Malik para $\Phi'(s)/s$, tomando $k = 1$: e^{-s^2/k^2} (color rojo) y $k^2/\left(k^2+s^2\right)$ (color azul)

Con este nuevo esquema no sólo se consigue respetar los bordes de la imagen, sino que incluso se ven realzados ligeramente. Para explicar este hecho, hay que observar lo siguiente. Definamos la función H como

$$H(s) = \frac{\Phi'(s)}{s}$$

donde $s = \|\nabla f_t\|$, con lo cual la ecuación 5.4 puede reescribirse como

$$2(f_t - f_0) - tdiv(H\nabla f_t) = 0 \tag{5.5}$$

El término de divergencia que aparece en esta ecuación puede expresarse como

$$div(H\nabla f_t) = (H + sH')\, f_{\eta\eta} + H f_{\xi\xi} \tag{5.6}$$

donde $f_{\eta\eta}$ y $f_{\xi\xi}$ son las segundas derivadas en las direcciones paralela y normal del gradiente respectivamente[1] (los detalles de la demostración pueden verse en [WEI98]).

Una forma de implementar numéricamente este esquema es mediante el método de descenso por gradiente. Ya que la ecuación 5.4 puede verse como la derivada del funcional de energía expuesto en la ecuación 5.3, el esquema iterativo siguiente nos iría acercando a una solución:

$$\begin{aligned} F_{n+1} &= F_n - \rho(2F_n - 2F_0 - t(H + sH')F_{\eta\eta} - tHF_{\xi\xi}) = \\ &= (1-2\rho)F_n + 2\rho F_0 + \rho t(H + sH')F_{\eta\eta} + \rho tHF_{\xi\xi} \end{aligned} \tag{5.7}$$

donde ρ controla la velocidad del descenso. Esto significa que, en cada iteración, el valor de cada pixel es una suma ponderada de cuatro términos: el valor actual del pixel F_n, el valor original F_0, y las segundas derivadas en las direcciones paralela y normal al gradiente en dicho pixel, $F_{\eta\eta}$ y $F_{\xi\xi}$. Estos dos últimos términos son los que producen una regularización en dichas direcciones, ya que el valor de la segunda derivada es proporcional a la diferencia entre el valor del punto y el de un entorno de dicho punto a lo largo de esa dirección.

Si tomamos por ejemplo la primera de las funciones propuestas por Perona-Malik, tenemos que

$$\begin{aligned} H &= e^{-s^2/k^2} \\ H + sH' &= \left(1 - 2\frac{s^2}{k^2}\right) H \end{aligned} \tag{5.8}$$

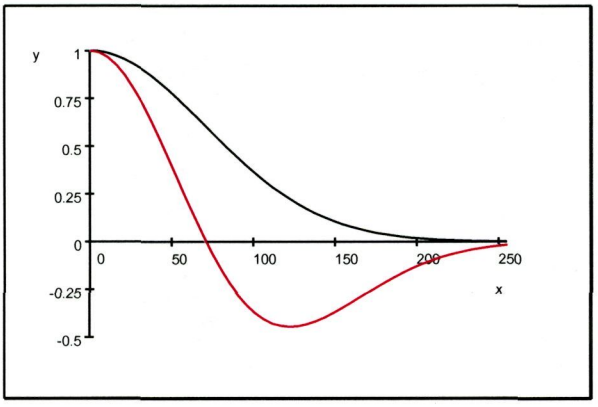


Figura 5.3: Función propuesta por Perona-Malik: $H = e^{-s^2/k^2}$ (color negro) y $H + sH'$ (color rojo), tomando $k = 100$

cuya gráficas se muestran en la figura 5.3.

Puede observarse como para valores bajos de gradiente, la difusión es alta en las dos direcciones, con lo cual se realiza una difusión parecida en ambas, lo que es similar al modelo lineal inicial, donde se utilizaba el laplaciano, ya que se cumple que

$$\Delta f = f_{xx} + f_{yy} = f_{\eta\eta} + f_{\xi\xi} \tag{5.9}$$

Sin embargo, para valores de gradiente más altos, la difusión en la dirección del borde (normal al gradiente) es más alta que en la dirección normal, llegando a ser negativo el término de difusión en la dirección del gradiente, lo que provoca el efecto de realce que comentábamos anteriormente.

Este efecto puede apreciarse mejor en la figura 5.4. Se ha escogido un círculo claro sobre fondo oscuro al cual se le ha añadido ruido, y se ha aplicado la formulación anterior, usando tres funciones H distintas. Hay que mencionar que inicialmente se ha suavizado la imagen con una gaussiana muy pequeña. El motivo del suavizado previo es para que la información de las derivadas esté un poco regularizada al inicio del proceso.

Figura 5.4: De izquierda a derecha: círculo de radio 100 con ruido añadido. Proceso tomando $H = 2$, $H = e^{-s^2/k^2}$, y $H = k^2/\left(k^2 + s^2\right)$. En todos los casos se ha tomado $t = 10$ y $k = 15$.

Se observa que tomando $H = 2$ estaríamos en el caso inicial, donde la difusión se produce por igual en todas las direcciones debido al laplaciano, por lo que los bordes aparecen totalmente difuminados. Sin embargo, para las funciones propuestas por Perona-Malik, la difusión en los bordes es mucho más pequeña.

[1] Expresado de otra forma, $f_{\eta\eta}$ y $f_{\xi\xi}$ son las segundas derivadas en las direcciones normal y paralela al contorno respectivamente.

Otras variaciones Como ya se ha comentado antes, la formulación de Perona-Malik fue sólo el comienzo de multitud de trabajos posteriores. Por reseñar algunos de ellas, Álvarez y otros [ALV92] propusieron modificar ligeramente la ecuación 5.1, la cual puede reescribirse también como

$$(f_t - f_0) - tdiv\,(\nabla f_t) = (f_t - f_0) - t\,(f_{\eta\eta} + f_{\xi\xi}) = 0$$

y eliminar el término $f_{\eta\eta}$ (paralelo al gradiente) de la expresión. Con ello controlaban que sólo hubiera difusión a lo largo de la línea del contorno, o más exactamente, a lo largo de las curvas de nivel de la función f. De esta forma no se mezclan las intensidades presentes a cada lado del borde.

Otra alternativa es el trabajo de Weickert [WEI94b], que cambia la función escalar $H(s)$ de la ecuación 5.5 por un tensor de difusión D, consistente en una matriz simétrica definida positiva. Este tensor produce que el flujo de la difusión no se produzca en la dirección del gradiente. Así, en lugar de minimizar la magnitud de la difusión en los bordes donde la norma del gradiente es alta (al ser H escalar sólo se modificaba la magnitud de la difusión, pero no su dirección), se altera también la dirección para realizar la difusión en la dirección del borde.

5.1.3 Problemas de la formulación

Con este tipo de esquemas se han conseguido muy buenos resultados en cuanto a la calidad visual de las imágenes restauradas. Sin embargo, nuestro estudio se centra en la exactitud de los parámetros del contorno sobre la imagen restaurada, y en este aspecto cabe reseñar varios detalles.

En primer lugar, en el esquema de la ecuación (5.7) aparecen dos derivadas direccionales, en las direcciones paralela y normal al gradiente en cada pixel. Ya hemos visto en capítulos anteriores que el cómputo de la dirección del gradiente en un pixel a partir de máscaras centradas en dicho pixel no es un proceso exacto, con lo cual, la dirección obtenida para cada pixel no será del todo correcta. Más aún, calcular una derivada de cualquier orden en una dirección diferente a los ejes principales también producirá un error, ya que sólo disponemos de valores discretos en una malla rectangular orientada en las direcciones x e y.

Por otro lado, el desarrollo matemático parte de que la imagen inicial f es continua y derivable en todo su dominio, lo que contradice nuestra hipótesis de trabajo. Además, al realizar la discretización de las expresiones se presupone que el valor de cada pixel indica el valor puntual de la función en ese punto, es decir

$$F(x_i, y_j) = f(x_i, y_j)$$

donde $F(x_i, y_j)$ es el pixel de coordenadas enteras (i, j), y $f(x_i, y_j)$ indica el valor de la función en dicho punto. Sin embargo, con nuestra segunda hipótesis, esto no es así, ya que partimos de que

$$F(x_i, y_j) = \frac{1}{h^2} \int_{y_j - h/2}^{y_j + h/2} \int_{x_i - h/2}^{x_i + h/2} f(x_i, y_j) dx dy$$

Esto significa que, como para estimar el valor de la derivada en un punto se está usando la expresión

$$\frac{\partial}{\partial x} f(x_i, y_j) \simeq \frac{F(x_i + h, y_j) - F(x_i - h, y_j)}{2h}$$

el error de esta aproximación será ligeramente mayor de lo previsto, si damos por cierta nuestra hipótesis.

Finalmente hay que reseñar que el valor de k en las funciones $\Phi(s)$ es vital para el resultado del proceso. Como puede apreciarse en la figura 5.3, interesa que k esté cercano al valor de la magnitud del gradiente que existe en los contornos que queremos resaltar. Por ejemplo, en dicha figura se ha tomado un valor $k = 100$. Por lo tanto, aquellos contornos con gradiente cercano a ese valor son los que mejor se restaurarán, ya que se difumina un poco en la dirección del contorno ($H > 0$), y se realza en la dirección normal ($H + sH' < 0$). Sin embargo, en contornos con gradiente menor, éstos tenderán a difuminarse en todas direcciones ($H \simeq H + sH'$), y en contornos con gradiente muy alto, apenas habrá difusión ($H \simeq 0$).

5.2 Método propuesto de restauración lineal

La idea principal subyacente en los métodos anteriores es siempre la misma: suavizar en aquellos pixels donde no existe contorno, y tratar de respetar los pixels en donde sí existe, bien realzando en la dirección normal al contorno, o bien suavizando a lo largo de la línea de éste. Nosotros proponemos un esquema que comparta esta idea, aprovechando los métodos propuestos en el capítulo anterior, de la forma que se muestra a continuación.

5.2.1 Algoritmo básico

Sea F_0 la imagen original (figura 5.5a), a la cual le aplicamos una máscara de suavizado H_1 para obtener la imagen suavizada G_0 (figura 5.5b). A esta nueva imagen le podemos aplicar nuestro algoritmo lineal para imágenes suavizadas (ver sección 4.6). Dicho algoritmo nos devolverá información sobre los pixels en que existe un contorno (figura 5.5c), así como sus parámetros intrapixel (posición, orientación y salto de intensidad). Con esta información, podría generarse una nueva imagen, F_1, tal que en las zonas alejadas de los pixels pertenecientes a un borde, mantengan el valor de la imagen suavizada, G_0, y en las zonas cercanas a los pixels borde, asignarles un valor calculado a partir de los parámetros obtenidos por el método para dicho pixel.

La forma para calcular estos valores se basa en la idea siguiente: sea el pixel (i, j) un pixel etiquetado por nuestro método como pixel borde. Esto significa que, a partir de los valores de una ventana 9x3 centrada en dicho pixel[2], nuestro método ha deducido que por él atraviesa una recta $y = a + bx$, dejando las intensidades A y B a cada lado de la recta. Por lo tanto, podría generarse una imagen sintética con las mismas dimensiones (9x3) a partir de los parámetros estimados por nuestro método. Esta idea puede verse en la figura 5.5d.

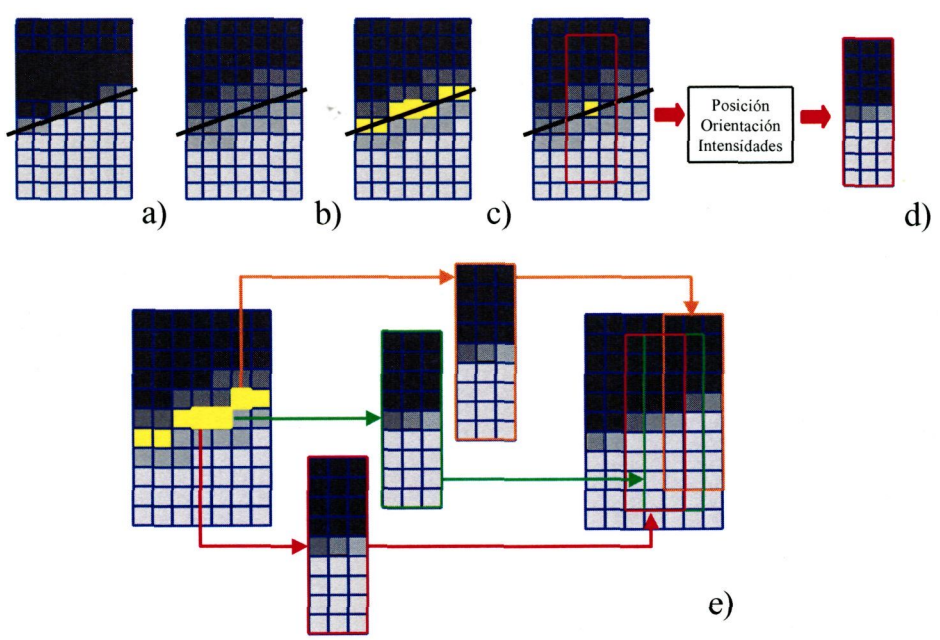


Figura 5.5: Método iterativo propuesto: a) Imagen original. b) Imagen suavizada. c) Pixels borde detectados por nuestro método. d) A partir de los parámetros detectados en cada pixel se genera una imagen sintética 9x3. e) Las ventanas sintéticas generadas en cada pixel borde se combinan para formar la imagen restaurada

[2] Una ventana 9x3 era la usada por nuestro método para los casos en que la pendiente está entre -1 y 1. Para los otros casos, la ventana sería 3x9

Para calcular la intensidad de cada uno de los 27 pixels de esa ventana, simplemente aplicamos nuestra hipótesis de partida pero a la inversa. Es decir, dado un pixel centrado en el punto (x_k, y_k), dada una recta $y = a + bx$, y dado dos valores de intensidad a cada lado del borde, A por debajo y B por encima, se calcula el área relativa interior al pixel bajo la recta, S, tal que $0 \leq S \leq 1$, y se deduce la intensidad que tendrá, $I = SA + (1 - S)B$.

Aplicando este esquema, en el caso en que la imagen original fuera ideal, obtendríamos una pequeña imagen sintética que coincidiría exactamente con la original. Pero si la imagen original tuviese algo de ruido, es entonces cuando mayor interés habría, ya que la pequeña imagen sintética generada seguiría dando como resultado los mismos parámetros para el contorno que la original, pero tendría una **calidad visual superior.**

Repitiendo esto para cada uno de los pixels borde, ocurrirá que a lo largo de un contorno, se iran generando diferentes subimágenes con solapes entre ellas. Por ejemplo, en la figura 5.5e, el pixel borde que aparece en el centro de la figura (remarcado en color verde) producirá una subimagen cuya columna derecha solapará con la columna central de la ventana generada para el pixel borde remarcado en violeta, y a su vez coincidirá con la columna izquierda de la ventana generada para el pixel borde remarcado en marrón. Para combinar toda la información y producir una única imagen F_1 proponemos el siguiente esquema:

Se crean dos imágenes de la misma resolución que la imagen original. Una de ellas es una imagen de contadores, C, donde el valor de cada pixel indica el número de ventanas que han sido generadas y en las que dicho pixel estaba incluido. La segunda imagen es una una imagen de intensidades, I, donde el valor de cada pixel indica la suma acumulada de las intensidades que se han ido calculando para cada pixel. Así, cuando hayamos procesado todos los pixels borde y generado todas las subimagenes necesarias, los pixels donde $C = 0$ indicarán que están lejos de cualquier borde, y por lo tanto tendrán el mismo valor que en la imagen suavizada G_0. Por el contrario, los pixels donde $C > 0$ indicarán que están incluidos en una o más ventanas sintéticas, y por tanto su color será el cociente I/C. El pseudo-código para este esquema es el siguiente, creado a partir de la función *ContornosSuavizados* de la página 142:

Funcion RealzarImagen (F0, F1, delta)

```
Crear una imagen de contadores C e inicializarla a 0
Crear una imagen de intensidades I e inicializarla a 0
Suavizar F con una mascara H1 para obtener la imagen G
Calcular las parciales Gx, Gy

Para todos los pixels (i,j) de la imagen G
   Si Gx(i,j)*Gx(i,j) + Gy(i,j)*Gy(i,j) < delta * delta Continuar
   Si |Gy(i,j)| > |Gx(i,j)| Entonces
      Si |Gy(i,j)| no es maximo en su columna Continuar
      ActualizarImagenes (C, I, i, j, vertical)
   Si No
      Si |Fx(i,j)| no es maximo en su fila Continuar
      ActualizarImagenes (C, I, i, j, horizontal)
   Fin Si
Fin Para

Crear una imagen F1 = G
Para todos los pixels (i,j) de la imagen G
   Si C(i,j) > 0 Entonces F1(i,j) = I(i,j) / C(i,j)
Fin Para
```

Funcion ActualizarImagenes (C, I, i, j, orientacion)

```
Segun orientacion vertical / horizontal:
Calcular parametros del contorno segun Lema 4.5 / 4.6
Para todos los pixels (m,n) pertenecientes a una ventanita (vert/hori) centrada en (i,j)
   Intensidad = Calcular intensidad del pixel (m,n) a partir
                                de los parametros calculados para (i,j)
   I(m,n) += Intensidad
   C(m,n) ++
Fin Para
```

El algoritmo anterior parte de una imagen original F_0 y produce una nueva imagen F_1 donde las zonas en las que no hay borde están suavizadas, y en las zonas cercanas a bordes, éstos aparecen más nítidos. La imagen F_1 tiene por tanto una calidad mayor, por lo cual le podríamos aplicar nuestro método para el cálculo de los parámetros y obtendríamos resultados más regulares. Esto puede verse en la figura 5.6.

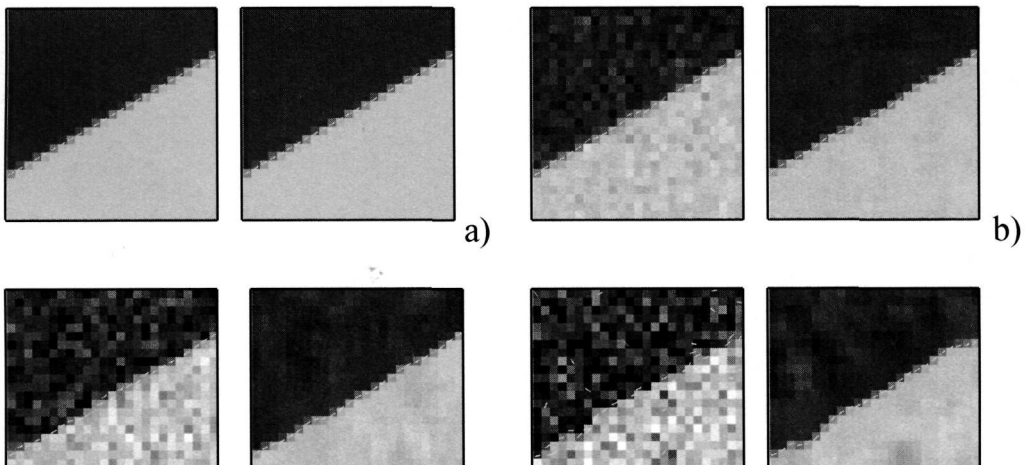


Figura 5.6: a) Contornos calculados sobre una rampa ideal, F_0, y sobre el resultado del algoritmo, F_1. b,c,d) igual pero a F_0 se le añadido ruido de magnitud 10, 20 y 30 respectivamente.

Un resultado muy importante es que cuando la imagen original es una rampa ideal (figura 5.6a), el método no altera la imagen, produciendo una imagen **idéntica a la original**. Esto no ocurre en ninguno de los métodos comentados en la sección anterior, ya que siempre se produce un pequeño difuminado al trabajar con los valores discretos de la imagen, que modifica los valores originales de los pixels. Discutiremos este resultado en una sección posterior.

En el caso en que el ruido presente en F_0 fuera muy grande (figura 5.6d), entonces F_1, aún siendo mejor que F_0, podría seguir teniendo bastante distorsión. En ese caso, podemos volver a aplicar el algoritmo a F_1 para obtener una nueva imagen F_2, y así sucesivamente para obtener F_n, imagen en la cual el cálculo de los contornos debería producir resultados mucho mejores que en la original. En la figura 5.7 se muestran resultados de varias iteraciones con imágenes de ruido mayor.

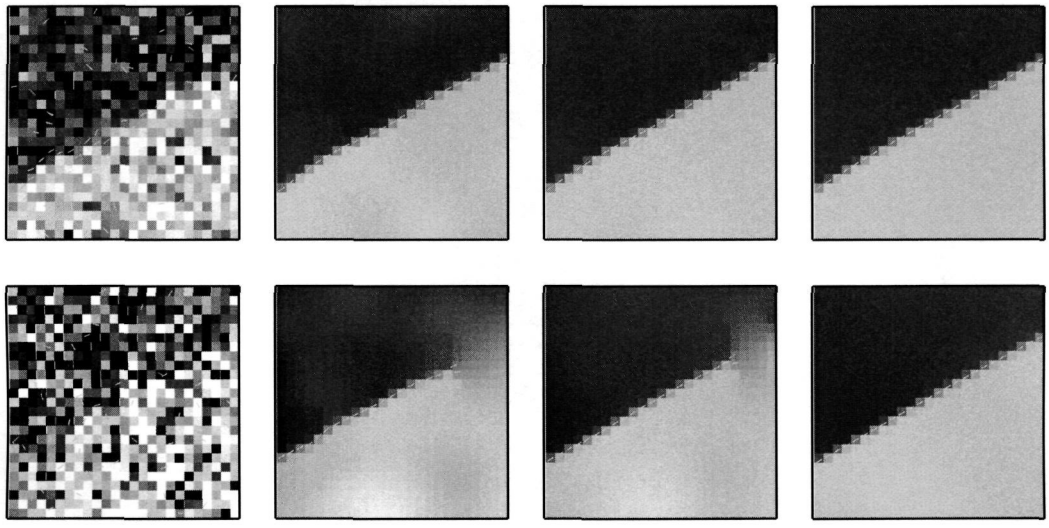


Figura 5.7: Rampa ideal con ruido. De arriba a abajo, la magnitud de ruido es 50 y 100. De izquierda a derecha, el número de iteraciones son 0, 10 20 y 50.

	Err. módulo		**Err. dirección**		**Err. posición**	
Iteraciones	Media	Max.	Media	Max.	Media	Max.
1000	0.81	1.41	1.10	3.32	22.2	41.3
2000	0.35	0.56	0.80	2.25	21.8	40.1
3000	0.14	0.23	0.67	1.75	21.7	40.5
4000	0.06	0.09	0.60	1.80	21.4	40.2
5000	0.02	0.04	0.55	1.82	21.1	39.7

Tabla 5.1: Resultados obtenidos por el método iterativo al aplicarlo a una rampa ideal con ruido 30

5.2.2 Aceleración de las iteraciones

Cuando se realiza un proceso iterativo sobre una imagen, es importante definir qué hacer en los márgenes (filas y columnas primera y última), ya que para los pixels de estas zonas no existe el mismo número de pixels vecinos que en el interior de la imagen, lo que impediría realizar los cálculos. Por ejemplo, colocar una ventana 9x3 centrada en un pixel de la primera columna de la imagen no tendría sentido, ya que dicha ventana no tendría columna izquierda. Más adelante defininiremos qué hacer, pero por ahora, aplicaremos el algoritmo sólamente en el interior de la imagen, dejando intacto los valores de los márgenes.

Vamos a medir el error cometido en los tres parámetros del contorno (intensidad, posición y orientación subpixel) en cada iteración, y veremos cómo se van reduciendo a medida que aumentan las iteraciones. Tomemos como imagen inicial una imagen ideal con una rampa de 30 grados, con intensidad 100 sobre el borde y 200 por debajo, a la que le hemos añadido un ruido de magnitud 30 (sólo en el interior, respetando los márgenes) (figura 5.6d). En la tabla 5.1 se muestra el error medio y máximo cometido en cada parámetro tras varias iteraciones.

Vemos que el error tiende a descender conforme avanzan las iteraciones, aunque el que mide la posición subpixel es el que más lento desciende. Después de 5000 iteraciones el error medio sigue siendo superior al 20% del tamaño del pixel. Para poder ver mejor cómo se van compensando los errores a lo largo del contorno podemos representar como abcisa de una gráfica el error de desplazamiento cometido en cada pixel del contorno tras

varias iteraciones (figura 5.8). Aunque en las primeras iteraciones los errores en cada pixel son independiente entre sí, éstos se van compensando entre pixels vecinos a medida que avanzan las iteraciones. Si se siguiera iterando indefinidamente, la curva se iría suavizando, y tendiendo a la horizontal.

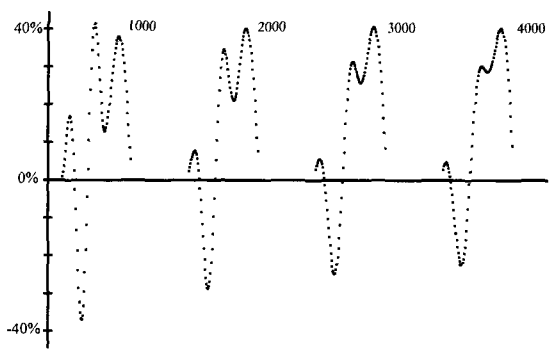


Figura 5.8: Error cometido en la posición sub-pixel después de varias iteraciones. Cada una de las curvas tiene tantos puntos como número de columnas tiene la imagen, y la altura de cada punto indica el porcentaje de error de la posición, relativo al tamaño del pixel. De izquierda a derecha: 1000, 2000, 3000 y 4000 iteraciones.

Técnica de aceleración propuesta Podemos acelerar el suavizado de esta señal aumentando el peso de la columna central de las subimágenes sintéticas que se van generando. Téngase en cuenta que cuando el método gaussiano obtiene un valor de posición y orientación para el contorno en un pixel concreto, dichos valores son para estimar el contorno en dicho pixel. Por lo tanto, cuando generamos la ventana 9x3, realmente la columna central es la que más relevancia tiene, y también la que menos error comete, puesto que cualquier pequeña variación en alguno de los parámetros haría que la intensidad calculada para las columnas laterales de la subimagen variase bastante.

Centrándonos en el ejemplo que estamos usando, donde la pendiente en valor absoluto es menor que 1 en todos los pixels del contorno, sabemos que nuestro método devuelve como borde un único pixel en cada columna. Por tanto, en cada columna i de la imagen se solapan tres subimagenes diferentes (las centradas en las columnas $i-1$, i e $i+1$). Tal y como hemos planteado en el algoritmo propuesto anteriormente, la intensidad calculada para la columna i será entonces un promedio de las tres, cuando realmente la que más importancia debería tener sería la centrada en la columna i.

Para solventar esto, podemos añadir un cierto peso (mayor que 1) a la columna central, quedando el bucle interior del algoritmo como sigue:

```
Intensidad = Calcular intensidad del pixel (m,n) a partir
                          de los parametros calculados para (i,j)
Si (m==i)
   I(m,n) += Peso * Intensidad
   C(m,n) += Peso
Si No
   I(m,n) += Intensidad
   C(m,n) ++
Fin Si
```

En la figura 5.9 se representa la misma señal de la figura 5.8 que mostraba el error de la posición tras 1000 iteraciones, pero para distintos valores de peso para la columna central. Puede verse cómo cuánto más grande sea este peso, más rápida es la convergencia.

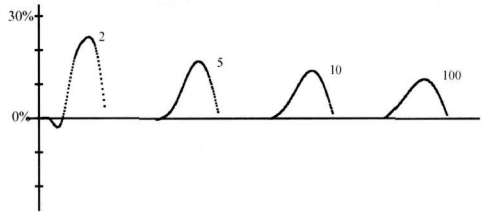


Figura 5.9: En cada una de las curvas se representa el error cometido en la posición sub-pixel después de 1000 iteraciones. De izquierda a derecha, el peso de la columna central en las subimágenes es 2, 5, 10 y 100.

Podría pensarse entonces que, si el peso es tan alto, debería entonces generarse solamente la columna central de la subimagen, es decir, generar ventanas de 9x1 en lugar de 9x3. Sin embargo, es interesante mantener las columnas laterales aún con menor peso. Una de las razones es que nuestro método detector de bordes es local a cada pixel, y no garantiza que todos los pixels borde localizados a lo largo de un mismo contorno formen una línea continua de pixels. Dicho de otra forma, en el ejemplo que nos ocupa, pudiera darse el caso que en alguna de las columnas, debido al elevado ruido existente, no se detectase ningún borde. Si sólo se generasen ventanas 9x1, esa columna se iría suavizando en cada iteración, ya que ninguna ventana caería en ella, lo que provocaría que fuese afectando al resto de columnas y finalmente toda la imagen se difuminará.

Sin embargo, al generar ventanas 9x3 con un peso muy pequeño en las columnas laterales, estamos diciendo que, si en una columna cae una ventana centrada en ella, ésta es la que decide la intensidad en dicha columna. Pero si en la columna no cae ninguna subimagen centrada, entonces se usará la intensidad estimada para una de las columnas laterales de la ventana. Es justo lo que ocurre en las imágenes de la fila inferior de la figura 5.7, donde pixels borde que inicialmente son suavizados **acaban por ser recuperados** tras varias iteraciones.

Una segunda razón es que a lo largo de un contorno la pendiente puede ir variando. Cada vez que atraviese el umbral de 45 grados, las ventanas pasarán de ser verticales a horizontales. Si sólo las generamos de grosor 1, el pixel indicado en color amarillo (ver figura 5.10) no pertenecería a ninguna ventana, por lo que se iría difuminando en cada iteración. Si el contorno tocase a ese pixel, también estaríamos cometiendo un error irrecuperable en la imagen.

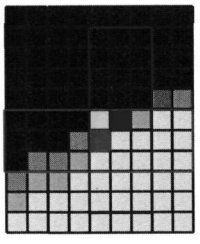


Figura 5.10: Situación de las ventanas generadas cuando un pixel (rojo) tiene un contorno con pendiente menor de 45 grados, y su vecino (verde) tiene pendiente mayor

A partir de ahora, le daremos un valor de 100 al peso de la columna central, con lo cual se pueden rehacer los cálculos de la tabla 5.1 y producir la tabla 5.2. En ella podemos ver que ahora los errores están más cerca

	Err. módulo		Err. dirección		Err. posición	
Iteraciones	Media	Max.	Media	Max.	Media	Max.
1000	0.80	1.43	0.10	0.23	6.81	11.5
2000	0.36	0.60	0.06	0.12	4.86	7.42
3000	0.16	0.26	0.04	0.07	3.39	5.17
4000	0.07	0.11	0.03	0.05	2.37	3.62
5000	0.03	0.05	0.02	0.03	1.66	2.53

Tabla 5.2: Resultados obtenidos por el método iterativo al aplicarlo a una rampa ideal con ruido 30

de cero tras las mismas iteraciones que antes, llegando a un error de posición medio por debajo del 2%.

5.2.3 Comparación con los métodos de restauración tradicionales

El método iterativo propuesto tiene varias ventajas con respecto a los comentados en la sección 5.1. Hay que recordar que el principal objetivo es lograr una mejor estimación de los parámetros del contorno usando la imagen resultante del proceso iterativo, y no simplemente restaurar la calidad visual de dicha imagen, cosa que sería más difícil de medir. Por lo tanto, lo que haremos en esta sección será usar el método iterativo propuesto y el de Perona Malik para obtener dos imágenes diferentes, y aplicar nuestro método detector de contornos a ambas imágenes para medir en cuál de ellas se obtiene un error menor con respecto a los verdaderos valores que tenía la imagen original.

Una imagen ideal permanece inalterable

La primera y muy importante característica de nuestro método de restauración es que cuando la imagen es una rampa ideal, se garantiza que $F_n = F_0$ para todas las iteraciones, sea cual sea el valor de la pendiente, intensidad y posición sub-pixel de la rampa, con lo cual los parámetros que se estimen para el contorno son siempre los exactos. Para los métodos de restauración tradicionales, esto no suele ser así en todos los casos. Veamos un ejemplo.

Cojamos una rampa ideal horizontal con un cierto desplazamiento subpixel, como la usada en la sección 1.3.2, que se ve en la figura 5.11, donde tenemos que la intensidad C es igual a

$$C = \frac{t}{h}A + \frac{h-t}{h}B$$

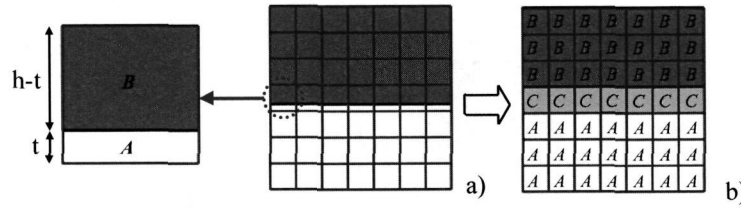


Figura 5.11: Borde horizontal con desplazamiento: a) Imagen original. b) Imagen digitalizada

Si le aplicamos un esquema de difusión anisotrópica como el de la sección 5.1.2, el resultado final será una imagen f_t que cumpla la ecuación 5.4. En un caso continuo parece claro que se va a cumplir, pero al aplicar un esquema discretizado veremos que eso no es así.

Para que en la imagen no se viese alterado el valor de ningún pixel, la expresión 5.6 de la divergencia debería valer cero en todos los pixels. Dicho de otra manera, en el esquema numérico de la ecuación 5.7, debería cumplirse que $F_{n+1} = F_n$ para todos los pixels. Si nos centramos en la primera iteración, $n = 0$, tenemos que

$$F_1 = F_0 + \rho t \left(H + sH'\right) F_{\eta\eta} + \rho t H F_{\xi\xi}$$

Sabemos que $F_{\xi\xi} = 0$ en toda la imagen, y que $F_{\eta\eta}$ también es nula excepto en las tres filas centrales. Por tanto, para que la imagen no varíe debe cumplirse que, al menos en esas tres filas,

$$H + sH' = 0$$

Sin embargo, no todas las funciones H posibles pueden garantizar esta condición. Si tomamos por ejemplo como función H la expresión 5.8, sólo funcionaría haciendo $k = \sqrt{2}s$. Pero eso es imposible ya que s indica el módulo del gradiente, y cada una de las tres filas centrales tiene un gradiente diferente, con lo cual alguna de las filas vería modificado su valor inicial.

Disminución efectiva del ruido

El comportamiento ante el ruido también es bastante efectivo con nuestro método. Si se continuase la tabla 5.2 durante más iteraciones, el error tendería a anularse completamente en los tres parámetros medidos[3]. Sin embargo, no ocurre lo mismo con el esquema de Perona-Malik, el cual alcanza la convergencia en errores medios cercanos a 4 unidades de intensidad para el módulo, 2 grados para la orientación y 9% de desplazamiento para la posición sub-pixel.

Incluso para ruidos de magnitud elevadísimos (mayores de 100), nuestro esquema sigue convergiendo a la imagen original, aunque cada vez con un mayor número de iteraciones necesario. En estos casos, aunque la convergencia a nivel numérico sea más lenta, a nivel visual ya se aprecia una mejora espectacular de la imagen con un número mucho menor de iteraciones. En la figura 5.12 se aprecia cómo en las 100 primeras iteraciones ya se aprecia una calidad en la imagen restaurada bastante buena, mientras que el esquema tradicional trata de seguir un compromiso entre suavizar muy fuerte para eliminar el ruido pero sin difuminar demasiado el contorno.

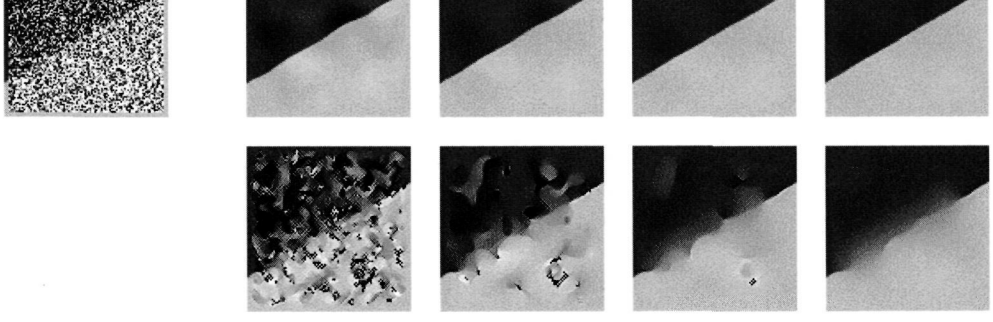


Figura 5.12: A la izquierda, rampa de 30 grados con ruido de magnitud 150. El resto de imágenes, la fila superior muestra el resultado de nuestro método, y la inferior el de Perona-Malik. De izquierda a derecha, 50, 100, 150 y 200 iteraciones.

[3] Recordemos que el ruido sólo ha sido añadido en el interior de la imagen, dejando intacto los márgenes de la imagen.

Autoenfoque

En la sección 4.8 vimos que cuando en una imagen existen contornos desenfocados, éstos se muestran en la imagen como si estuviesen ligeramente difuminados. Si el desenfoque no es muy fuerte, nuestro método era capaz de seguir encontrando los parámetros del contorno con una precisión muy fina. Esto significa que, como nuestro método iterativo va generando pequeñas ventanas sintéticas a partir de los parámetros obtenidos por el método detector, ya desde la primera iteración el efecto de desenfoque desaparecería, y aparecería un contorno nítido tal y como presupone nuestra hipótesis de trabajo.

Por ejemplo, en la figura 5.13 se ha tomado la rampa ideal de 30 grados, y se ha suavizado con una máscara de radio 2 para simular un desenfoque. Vemos que visualmente se consigue casi la imagen original. Numéricamente hay pequeños errores debido a que en algunos pixels la franja difuminada del contorno es mayor que el alto de la ventana sobre la que estimamos los parámetros en cada pixel (ver figura 4.24). Por otro lado, si además a la imagen desenfocada se le añade ruido, el método sigue recuperando el contorno con una calidad muy superior al de Perona Malik.

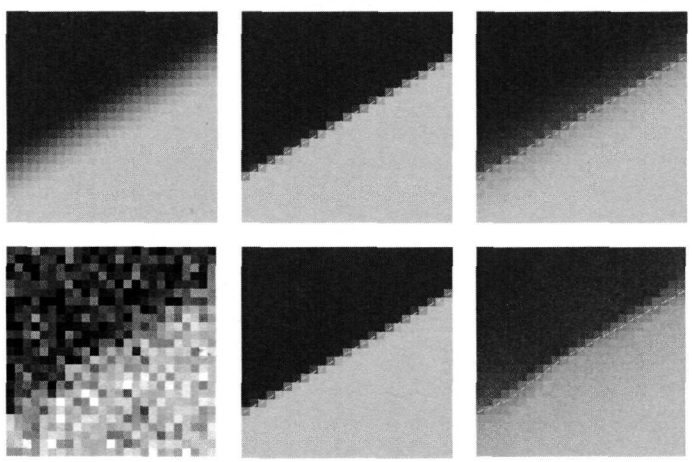


Figura 5.13: Fila superior, de izquierda a derecha: rampa de 30 grados suavizada con una máscara de radio 2; resultado con nuestro método y con Perona-Malik. Fila inferior: igual, pero añadiendo ruido de magnitud 30 a la imagen suavizada.

Esta característica puede apreciarse muy bien en la imagen del capítulo anterior (figura 4.23), donde aparecían un perchero y varios objetos circulares desenfocados. Aplicando nuestro método del capítulo anterior obteníamos las imágenes de la figura 4.25. Sin embargo, si aplicamos unas pocas iteraciones de nuestro método iterativo, obtenemos una imagen como la de la figura 5.14 con sus contornos perfectamente definidos. Si a esta nueva imagen se le detectan los contornos, el resultado es aún mejor.

Robustez frente a distintos niveles de ruido y de intensidad

Para conseguir todas las imágenes de Perona-Malik que se han mostrado hubo que buscar valores óptimos para los coeficientes t (que mide el compromiso entre conservar la imagen original y minimizar el gradiente) y k (que indica el umbral de gradiente de lo que consideramos un contorno). Aun así, si en una misma imagen hubiera contornos con distintos saltos de intensidad y niveles de ruido, donde algunos estuvieran desenfocados y otros no, no podría obtenerse un resultado igual de bueno para todos ellos.

Por el contrario, nuestro método tiene un único parámetro que indica el umbral de intensidad δ. Dicho umbral no debe ser muy alto, ya que los contornos cuyo salto de intensidad sea inferior a δ quedarán difuminados en

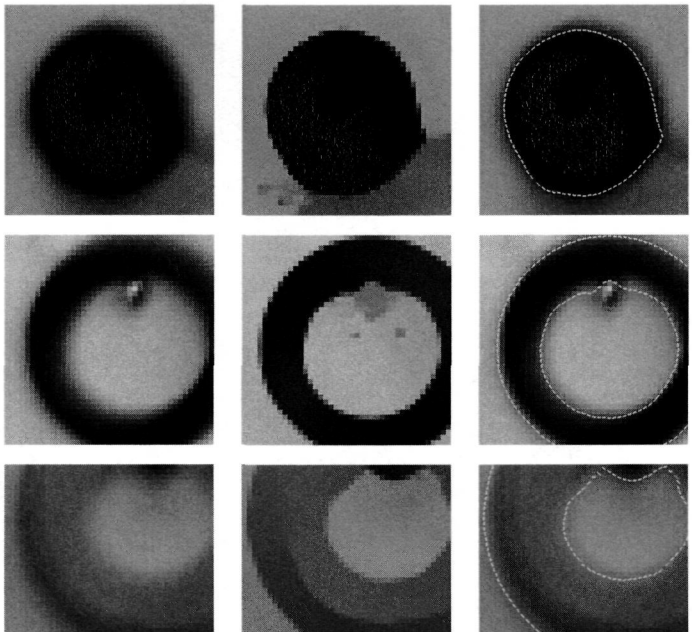


Figura 5.14: De izquierda a derecha: imagen original, imagen restaurada, y contornos estimados sobre esta última, visualizados sobre la imagen original

el proceso. Sin embargo, cuando el umbral es bajo, no es tan importante el valor concreto que tenga, ya que el resultado final será bastante parecido. Esto puede verse en la figura 5.15, donde se muestra el resultado de la primera iteración, F_1, para la rampa con ruido que venimos usando. En la parte derecha se ha usado un umbral de 50 unidades, con lo cual en la zona que se muestra ampliada no se detecta ningún pixel borde, y la imagen mostrada coincide con el suavizado hecho a la imagen original F_0.

En la parte central, el umbral es de 20 unidades, con lo cual aparecen algunos puntos etiquetados como borde en la zona ampliada. En la figura puede verse además claramente la ventana que se ha generado para el pixel señalado en rojo. Como son ventanas aisladas, en la siguiente iteración que se hiciese, después de hacer el suavizado siguiente, el salto de intensidad será menor, y así sucesivamente en cada iteración, llegando un momento en que el valor estuviese por debajo del umbral, con lo que toda esa zona acabaría difuminándose igualmente.

Finalmente en la parte izquierda, donde el umbral es de 10 unidades, se detectan multitud de puntos borde, con lo cual se generan muchas ventanas sintéticas. Eso significa que existen muchos pixels que están incluidos en más de una ventana. Al tratarse de una zona homógenea con ruido, las distintas ventanas que incluyan un mismo pixel serán bastante diferentes entre sí, al contrario de lo que ocurriría a lo largo de un contorno. Así, al generar la intensidad de ese pixel, éste tendrá un valor promediado de su entorno. Y si continuásemos con las iteraciones, toda esa zona acabaría difuminándose como en el caso anterior. Es por ello que puede afirmarse que después de muchas iteraciones, el resultado es el mismo para diferentes valores del umbral.

Todo lo anterior significa una menor sensibilidad al valor del umbral decidido, al contrario que ocurría con el esquema Perona-Malik. Esto nos permite que si en una misma imagen existen contornos con distintos saltos de intensidades, y algunos están más desenfocados que otros, no tenemos que preocuparnos por establecer un valor óptimo, sino simplemente dar un valor inferior al del contorno menos resaltado que queramos localizar.

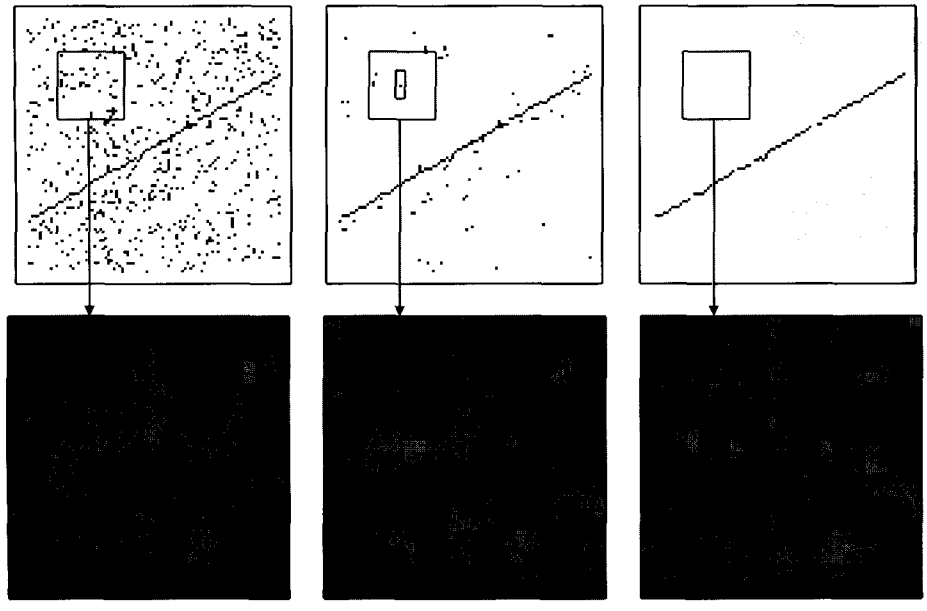


Figura 5.15: Puntos encontrados como borde tras una sola iteración de nuestro método, para distintos umbrales (de izquierda a derecha, 10, 20 y 50).

Esto puede apreciarse en la figura 5.16, donde aparecen dos contornos con saltos de intensidad 40 y 80 unidades, y uno de ellos está desenfocado (ha sido suavizado con una máscara de radio 2). Además, a la mitad de la derecha se le ha añadido un ruido de magnitud 50. Vemos cómo nuestro método restaura de forma óptima los dos contornos, al contrario que el esquema de Perona.

5.2.4 Tratamiento de los márgenes

Hasta ahora habíamos dejado intactos los márgenes de la imagen, pero para poder aplicarlo a imágenes reales, es necesario que adoptemos alguna estrategia, primero en la forma de suavizar, y posteriormente en el método de detección de contornos y en la generación de ventanas sintéticas.

Suavizado

Como el suavizado se está realizando con una máscara de radio 1, aparece una incongruencia en las filas primera y última (y también en las columnas). Una solución para estos pixels podría ser promediar solamente con sus 5 pixels vecinos, en lugar de con los 8 vecinos como ocurre en un pixel del interior (ver figura 5.17a). Otra solución alternativa podría ser suponer que la imagen es contante fuera de sus márgenes, es decir, que existiría una columna y una fila "virtual" por fuera de cada uno de los márgenes de la imagen, cuyo valor coincide exactamente con los pixels de cada margen (figura 5.17b).

Sin embargo, cuando se trata de un contorno que toca el margen, ninguna de estas alternativas es aconsejable, y es mejor elegir una tercera opción. Tomemos como ejemplo el contorno de la figura 5.17 que corta el margen izquierdo de la foto. De no haber existido dicho margen, o de haber estado varias columnas más a la izquierda, parece lógico suponer que la línea continuaría con una pendiente similar a la que tiene cuando alcanza el margen. Por lo tanto, a la izquierda de la columna del margen habría otra columna cuyos pixels tendrían una intensidad

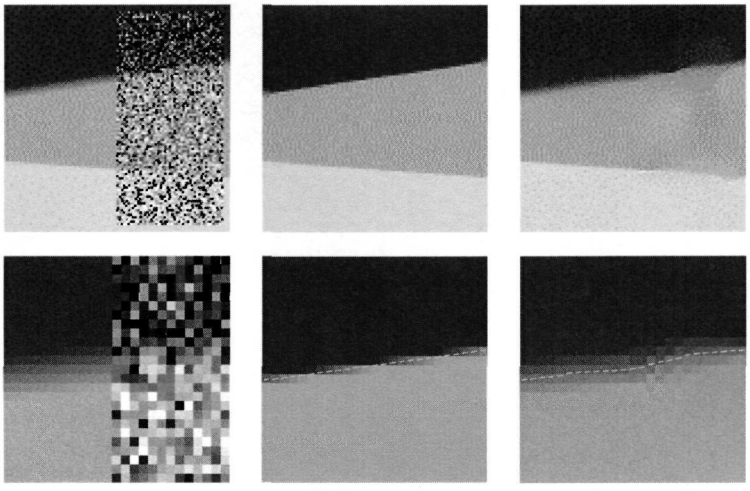


Figura 5.16: Fila superior (de izquierda a derecha): imagen original, y resultados de nuestro método y del de Perona-Malik. Fila inferior: detalle ampliado de cada imagen.

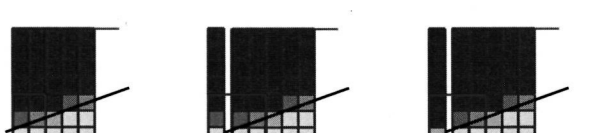


Figura 5.17: Alternativas para suavizar el margen izquierdo de una imagen: a) el pixel se promedia sólamente con sus cinco vecinos; b) se crea una columna adicional más igual a la columna del margen; c) se crea una columna adicional más con intensidades proporcionales a la recta del contorno.

acorde con la posición de la recta en dicha columna (figura 5.17c), y que podría ser incluida dentro del proceso de suavizado.

Pero el problema es que para poder estimar entonces las intensidades de los pixels de esa nueva columna deberíamos conocer los parámetros del contorno en el margen, y eso aún no lo conocemos, ya que precisamente para eso es para lo que queremos suavizar la imagen. Sin embargo, hay algo que sí podemos hacer. Supongamos que dicho contorno se corresponde localmente como una línea recta. Y llamemos S_L, S_M y S_R a las sumas de los valores de los pixels de las columnas izquierda, central y derecha de vecinos del pixel respectivamente dentro de la máscara. Por tanto, ya que las intensidades de los pixels son proporcionales a las áreas bajo la recta interiores a cada pixel, podemos asumir que, en la mayoría de los casos[4],

$$S_M - S_L \simeq S_R - S_M$$

[4]Siempre y cuando la recta corte a la ventana 3x3 de vecinos de izquierda a derecha.

con lo cual, podemos afirmar que

$$F * H_1 = F * \begin{pmatrix} a_{11} & a_{01} & a_{11} \\ a_{01} & a_{00} & a_{01} \\ a_{11} & a_{01} & a_{11} \end{pmatrix} \simeq F * \frac{1}{a_{00} + 2a_{01}} \begin{pmatrix} a_{01} \\ a_{00} \\ a_{01} \end{pmatrix}$$

Es decir, para suavizar el margen izquierdo (y derecho), sólo tendremos en cuenta los pixels vecinos en esa misma columna, y no en las columnas adyacentes. Para las filas la conclusión es similar, pero análoga. Así los contornos se podrían mantener estables durante un mayor número de iteraciones.

Esquinas Las cuatro esquinas de la imagen son un caso aún más especial, y realmente tienen muy poco peso en la totalidad de la imagen. Pero basándonos en la idea anterior podemos usar la siguiente convolución (para el caso del pixel de la esquina superior izquierda):

$$F * \begin{pmatrix} a_{11} & a_{01} & a_{11} \\ a_{01} & a_{00} & a_{01} \\ a_{11} & a_{01} & a_{11} \end{pmatrix} \simeq F * \frac{1}{2a_{00} + 2a_{01}} \begin{pmatrix} 2a_{00} & a_{01} \\ a_{01} & 0 \end{pmatrix}$$

y de forma simétrica para las otras tres esquinas.

Detección de contornos y generación de subimágenes

Una vez tenemos la imagen suavizada, aplicaremos el detector de contornos propuesto para saber en qué pixels generar las subimágenes. Tomemos como ejemplo un contorno recto de pendiente menor que 1, como el de la figura 5.18a. Ya que las subimágenes que generamos son de resolución 9x3, no haría falta detectar nada en la columna del margen izquierdo. Simplemente haríamos la estimación para la segunda columna de la imagen, en donde dispondríamos de sus 8 vecinos en la imagen suavizada, y generaríamos la ventana 9x3 la cual caería sobre la columna del margen, generando nuevos valores para la siguiente iteración. Para el margen derecho se procede de la misma manera. Esto significa que el bucle central de nuestro algoritmo solo detectaría contornos en el interior de la imagen.

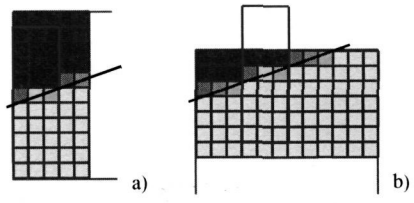


Figura 5.18: Situación cuando un contorno de pendiente menor que 1 corta un margen: a) horizontal. b) vertical.

Sin embargo, para contornos de pendiente menor que 1, la detección cuando éstos cortan un margen horizontal (superior o inferior) es totalmente diferente, ya que la vecindad 9x3 necesaria para realizar la estimación no puede ubicarse (figura 5.18b). En cualquier caso, alguna ventana habrá que generar en esos pixels, puesto que de lo contrario se irían difuminando en cada iteración, y traspasando este efecto al resto del contorno. La idea que proponemos es, en el caso de la figura en donde el contorno toca el margen superior, estimar solamente el valor de la intensidad bajo el contorno (eso sí puede hacerse porque la mitad inferior de la ventana está entera) y generar una ventana con esa intensidad, dejando el resto de pixels igual que en la imagen original.

En la figura 5.19 puede verse como el resultado es satisfactorio para los dos tipos de cortes con los márgenes, incluso en el caso de existencia de ruido. En la parte derecha puede verse como el ruido no desaparece del todo en la zona pegada al margen superior, pero al menos no afecta al resto del contorno, que era el objetivo, puesto que lo lógico es asumir que los contornos cuyos parámetros quieren estimarse con mayor precisión estarán en el interior de la imagen, y no cortando los márgenes.

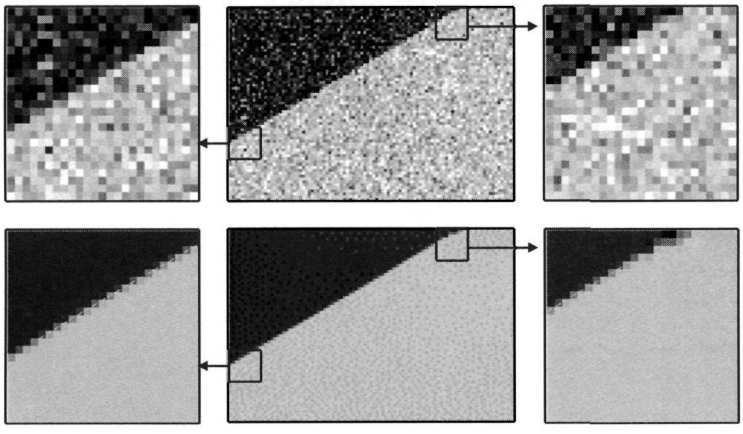


Figura 5.19: Fila superior: contorno de 30 grados que corta los márgenes izquierdo y superior de la imagen, a la que se ha añadido ruido de magnitud 30. Fila inferior: resultado de nuestro método.

Numéricamente el error no tiende a hacerse cero, como ocurría en el caso donde los márgenes de la imagen no se tocaban, ya que precisamente en estas zonas es donde no pueden estimarse los parámetros del contorno con total precisión. Sin embargo, en el interior de la imagen, el error es insignificante.

5.3 Método propuesto de restauración cuadrático

Hasta ahora todas las pruebas de restauración se han hecho con contornos rectos. ¿Qué ocurre cuando son curvos? El sistema también da estimaciones correctas, pero tiende a suavizar demasiado los contornos a medida que avanzan las iteraciones. Por ejemplo, tomemos un círculo ideal de radio 20 y apliquémosle nuestro método lineal. En la figura 5.20 puede verse el resultado tras varias iteraciones: el círculo se va haciendo pequeño hasta desaparecer.

La explicación es la siguiente: vimos en la sección 4.4 que, para un pixel borde con un contorno de pendiente menor que 1, el método lineal encontraba la expresión de la recta $y = a' + b'x$, mientras que el método cuadrático encontraba la parábola $y = a + bx + cx^2$, donde la relación entre los coeficientes de ambos métodos era

$$\begin{aligned} a &= a' - \lambda c \\ b' &= b \end{aligned}$$

donde $\lambda > 0$ dependía de los coeficientes de la máscara de suavizado (ecuación 4.13). Es decir, si asumimos que el contorno viene dado por una expresión de segundo grado, al aproximarlo por una recta estamos cometiendo un error en la posición proporcional al coeficiente de segundo grado (figura 5.21a). Dicha posición viene medida en la vertical central del pixel ($x = 0$). En ese punto, la curvatura es

$$K = \frac{2c}{(1 + b^2)^{3/2}}$$

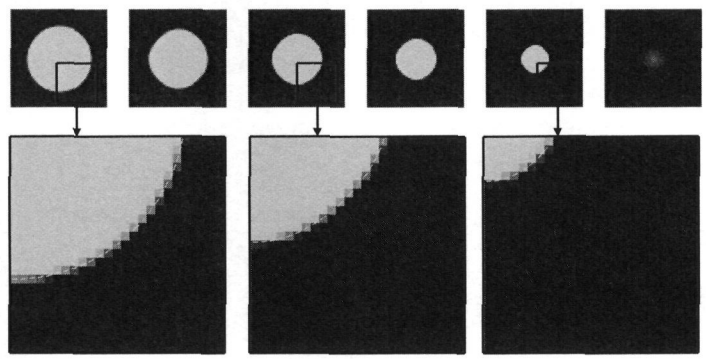


Figura 5.20: Resultado de aplicar el método lineal a un círculo de radio 20, tras 100, 200, 300, 400 y 500 iteraciones.

Por tanto, el error cometido en la posición es

$$\text{error} = \frac{\lambda}{2R}\sqrt{(1+b^2)^3}$$

donde R es el radio de curvatura. Por ejemplo, para el caso del círculo de radio 20, y empleando una máscara de suavizado con todos los coeficientes iguales entre sí ($\lambda = 3/4$), el error cometido para los distintos valores de pendiente (entre 0 y 1) se muestran en la figura 5.21b. No son errores demasiado grandes (menos de un 5% del tamaño del pixel en las diagonales), pero se van acumulando en cada iteración. Atendiendo al signo, el error se produce siempre hacia el interior del círculo, con lo que cada ventana que se genera va acercando el contorno hacia el centro, hasta hacerlo desaparecer.

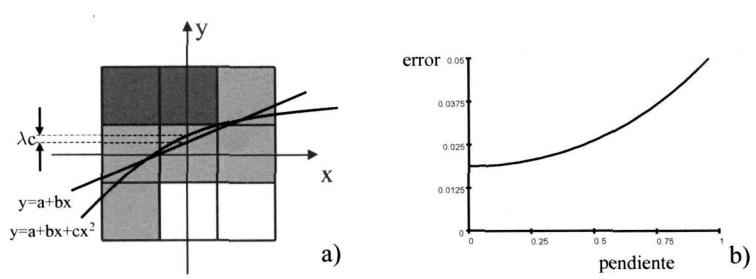


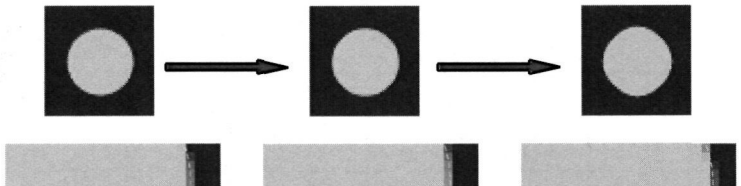
Figura 5.21: a) diferencia de posición entre el método detector lineal y cuadrático. b) error cometido en la posición sub-pixel por el método detector lineal para estimar la posición de un círculo de radio 20, en función de la pendiente del contorno.

5.3.1 Algoritmo básico

Para arreglar el problema descrito anteriormente y no perder la forma original del contorno seguiremos usando el mismo algoritmo de la sección anterior, sustituyendo el método de detección lineal por el de segundo grado. Todo lo demás (mayor peso en la columna central y tratamiento de los márgenes) sigue exactamente igual.

Simultáneamente, habrá que modificar el proceso de generación de las ventanas para poder representar curvas de segundo grado, en lugar de rectas. Las ventanas siguen teniendo las mismas dimensiones que antes (9x3 para contornos de pendiente entre -1 y 1), y la diferencia es que ahora, para calcular la intensidad de cada uno de los 27 pixels de la ventana, una vez estimados los valores de los coeficientes a, b y c y las intensidades A y B, habrá que calcular el área relativa interior al pixel bajo la parábola, S, tal que $0 \leq S \leq 1$, y deducir su intensidad de igual manera que antes mediante la expresión $I = SA + (1 - S)B$.

Si aplicamos ahora este algoritmo a un círculo de radio 15 (figura 5.22), el círculo ya no desaparece como antes, pero a medida que avanzan las iteraciones su forma se va perdiendo, sobre todo por las diagonales. ¿A qué es debido este efecto? Recordemos que la idea central de nuestro método detector es aproximar localmente el contorno por una curva de segundo grado (una parábola), y usar la posición, pendiente y curvatura del punto sobre la vertical central del pixel ($x = 0$) para dar una estimación de las características del contorno dentro de la curva (figura 5.23a). Por tanto, será más exacto estimar una circunferencia a partir de estas características, y usarla para generar las ventanas sintéticas, en lugar de la parábola (figura 5.23b).



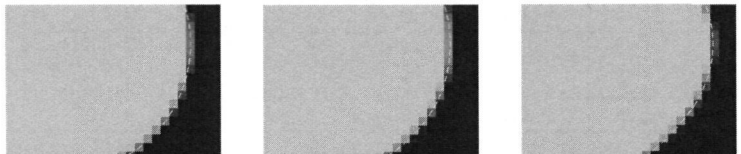


Figura 5.22: Resultado de aplicar el método cuadrático a un círculo de radio 15 tras 1000 y 5000 iteraciones.

Generación de circunferencias El hecho de usar circunferencias permite una mayor invarianza a rotaciones del contorno en la imagen, ya que la parábola, al contrario de la circunferencia, no mantiene la misma curvatura en todos sus puntos. Además, no varía igual en zonas próximas a la pendiente horizontal, que cuando nos acercamos a pendientes de 45^o. Por ejemplo, sea una circunferencia de radio R que pasa por el origen con pendiente 0. La parábola que comparte posición, pendiente y curvatura en ese punto es

$$y = -\frac{1}{2R}x^2$$

En un entorno de 3x3 pixels (figura 5.23c) apenas hay diferencias entre ambas curvas. Sin embargo, si la misma circunferencia tocase el origen con pendiente 1, la parábola sería

$$y = x - \frac{\sqrt{2}}{R}x^2$$

y en la figura 5.23d puede apreciarse la pequeña diferencia de áreas entre ambas. La circunferencia se cierra mucho más rápido que la parábola conforme va descendiendo, y mientras la primera es simétrica en ambas orientaciones, la parábola adquiere curvaturas diferentes conforme se aleja de su vértice. Y es precisamente este hecho el que provoca la deformación en las diagonales de la figura 5.22. Por lo tanto, generar circunferencias en lugar de parábolas producirá contornos más similares independientemente de la orientación que tengan dentro de la imagen.

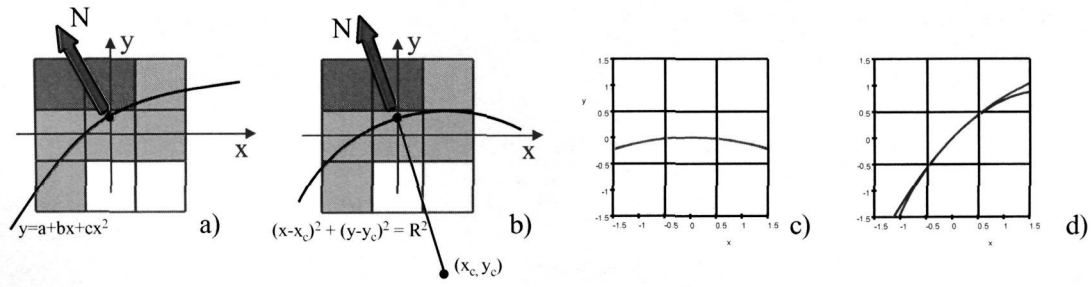


Figura 5.23: a) El punto de la vertical central se usa para estimar los parámetros del contorno en el interior del pixel. b) Aproximar por arcos de circunferencia sería más preciso. c) Parábola y circunferencia de radio 5 que comparten igual posición, pendiente y curvatura en el origen (pendiente 0). d) Igual pero con pendiente 1.

	Err. módulo		**Err. dirección**		**Err. posición**		**Radio de curvatura**		
Iteraciones	Media	Max.	Media	Max.	Media	Max.	Media	Min.	Max.
1000	0.0	0.0	0.46	0.88	21.6	24.1	15.0	13.7	17.3
2000	0.0	0.0	0.90	1.68	34.2	40.2	15.0	13.9	18.9
3000	0.0	0.0	1.54	2.49	43.9	57.0	15.0	13.5	20.4
4000	0.0	0.0	2.01	3.11	51.6	70.1	15.0	12.7	21.5
5000	0.0	0.0	2.35	3.63	57.8	80.3	15.0	12.2	22.5

Tabla 5.3: Errores cometidos por el método cuadrático en un círculo de radio 15.

Para estimar la circunferencia que comparte posición, pendiente y curvatura con la parábola en el punto $x = 0$ hay que hacer los siguientes cálculos, atendiendo a la figura 5.23b. El punto en $x = 0$ tiene de coordenadas $(0, a)$, vector normal $[b, -1]$ y curvatura $2c/(1+b^2)^{3/2}$. De aquí podemos deducir que el radio de la circunferencia será

$$R = \frac{\left(1+b^2\right)^{3/2}}{2\,|c|} \tag{5.10}$$

y que el centro está en la posición

$$\begin{aligned} x_c &= R\frac{b}{\sqrt{1+b^2}} = -\frac{b\left(1+b^2\right)}{2c} \\ y_c &= a - R\frac{1}{\sqrt{1+b^2}} = a + \frac{1+b^2}{2c} \end{aligned} \tag{5.11}$$

Si ahora aplicamos de nuevo el método a la imagen del círculo de radio 15, visualmente la imagen no parece variar, pero numéricamente no existe convergencia hacia los valores reales, sino que se producen pequeños errores en los parámetros, que se van incrementando lentamente conforme avanzan las iteraciones (ver tabla 5.3). En un caso práctico, estos errores pueden ser insignificantes, puesto que normalmente no hará falta un número tan alto de iteraciones. Sin embargo, a nivel teórico, el objetivo debería seguir siendo el mantener inalterable el valor de los pixels de una imagen ideal de una circunferencia. En la siguiente sección proponemos una optimización al algoritmo para lograrlo.

5.3.2 Algoritmo optimizado

La principal causa de los errores es debido a que nuestro método detector aproxima la curva del contorno mediante parábolas en lugar de circunferencias. Eso significa que para lograr nuestro objetivo no basta con generar circunferencias en las ventanas, sino que hay que detectarlas con precisión. Ya expusimos en la sección 3.3.4 que para estimar la curvatura local del contorno sería más correcto aproximar la curva por un arco de circunferencia, puesto que, a diferencia de una parábola, es invariante a rotaciones. De esta manera, podría detectarse la curvatura de un punto del contorno con la misma precisión, independientemente de su orientación en la imagen.

Pero el principal problema de buscar directamente la circunferencia a partir de las intensidades de los pixels de la ventana es que da lugar a un sistema de ecuaciones no lineal, complejo de resolver. Como en los capítulos anteriores el interés era simplemente dar una estimación de los contornos, el error cometido por usar parábolas era despreciable. Sin embargo, al plantear un esquema iterativo donde cualquier pequeño error puede propagarse, se hace necesario una mayor precisión.

Error cometido

Analicemos cuál es el error exacto que comete nuestro método cuando la curva del contorno es circular. Recordemos de la sección 4.4 que el método partía de que las áreas Q_i, P_i y T_i venían dadas por las expresiones

$$\begin{aligned} P_i &= \int_{(i-\frac{1}{2})h}^{(i+\frac{1}{2})h} \left(y(x) + \frac{5}{2}h \right) dx \\ Q_i &= \int_{(i-\frac{1}{2})h}^{(i+\frac{1}{2})h} \left(y(x) + \frac{7}{2}h \right) dx = P_i + h^2 \\ T_i &= \int_{(i-\frac{1}{2})h}^{(i+\frac{1}{2})h} \left(y(x) + \frac{3}{2}h \right) dx = P_i - h^2 \end{aligned}$$

donde $y(x)$ representaba la curva del contorno, la cual se asumía que localmente cumplía la expresión $a + bx + cx^2$.

Supongamos ahora que la expresión $y(x)$ de un contorno real es desconocida, y llamemos $U(x)$ a su función primitiva, la cual también es desconocida, tal que $U'(x) = y(x)$. Entonces,

$$P_i = \left[U(x) + \frac{5}{2}hx \right]_{(i-\frac{1}{2})h}^{(i+\frac{1}{2})h} = U_{ih+h/2} - U_{ih-h/2} + \frac{5}{2}h^2 \tag{5.12}$$

Si se rehiciese todo el desarrollo del método tal y como se hizo en la sección 4.4, obtendríamos las siguientes expresiones[5] para las sumas de las columnas

$$\begin{aligned} S_L &= \frac{5}{2}A + \frac{9}{2}B + \frac{1}{3}(A-B)\left(U_{5/2} - U_{-1/2}\right) \\ S_M &= \frac{7}{2}(A+B) + \frac{1}{3}(A-B)\left(U_{3/2} - U_{-3/2}\right) \\ S_R &= \frac{9}{2}A + \frac{5}{2}B + \frac{1}{3}(A-B)\left(U_{1/2} - U_{-5/2}\right) \end{aligned}$$

[5] Para hacer más sencillas las expresiones, se ha tomado el caso $h = 1$ y $a_{00} = a_{01} = a_{11} = 1/9$.

El resultado de aplicar nuestro método detector daría la parábola estimada $P_E(x) = a_E + b_E x + c_E x^2$, donde los coeficientes tendrían las expresiones

$$\begin{aligned} c_E &= \frac{S_L + S_R - 2S_M}{2(A-B)} = \frac{1}{6}\left(U_{5/2} + U_{1/2} - U_{-1/2} - U_{-5/2}\right) - \frac{1}{3}\left(U_{3/2} - U_{-3/2}\right) \\ b_E &= 1 + \frac{S_R - S_L}{2(A-B)} = \frac{1}{6}\left(U_{5/2} - U_{1/2} - U_{-1/2} + U_{-5/2}\right) \\ a_E &= \frac{2S_M - 7(A+B)}{2(A-B)} - \frac{3}{4}c = \frac{7}{12}\left(U_{3/2} - U_{-3/2}\right) - \frac{1}{8}\left(U_{5/2} + U_{1/2} - U_{-1/2} - U_{-5/2}\right) \end{aligned} \tag{5.13}$$

Es fácil comprobar que si realmente el contorno $y(x)$ sigue la expresión $a + bx + cx^2$, su función primitiva tiene la expresión

$$U(x) = ax + \frac{b}{2}x^2 + \frac{c}{3}x^3$$

y las expresiones anteriores dan como resultado exactamente los coeficientes originales de la parábola.

Supongamos ahora que el contorno $y(x)$ sigue la expresión de una circunferencia[6], es decir,

$$y(x) = y_c + \sqrt{R^2 - (x - x_c)^2}$$

donde (x_c, y_c) es el centro de la circunferencia, y R el radio. En este caso, la función primitiva viene dada por la expresión

$$U(x) = y_c x + \frac{R^2}{2}\arcsin\left(\frac{x - x_c}{R}\right) + \frac{x - x_c}{2}\sqrt{R^2 - (x - x_c)^2} \tag{5.14}$$

Si ahora evaluamos nuestro método, el resultado será una expresión bastante compleja para cada uno de los tres coeficientes, que no va a ser además el resultado deseado. ¿Cuál sería éste? Pues lo ideal sería obtener aquella parábola que, medida sobre la vertical central del pixel ($x = 0$), compartiese el mismo valor para la posición, pendiente y curvatura que la circunferencia (figura 5.23b). Analíticamente podemos calcular dichos valores:

$$\begin{aligned} \text{corte con la vertical central del pixel:} &\quad \left(0, y_c + \sqrt{R^2 - x_c^2}\right) \\ \text{vector normal:} &\quad N = \frac{A-B}{R}\left[\, -x_c, \quad \sqrt{R^2 - x_c^2}\,\right] \\ \text{curvatura} &: \quad K = \frac{1}{R} \end{aligned}$$

Por tanto, la parábola ideal sería $P_I(x) = a_I + b_I x + c_I x^2$, donde

$$\begin{aligned} a_I &= y_c + \sqrt{R^2 - x_c^2} \\ b_I &= \frac{x_c}{\sqrt{R^2 - x_c^2}} \\ c_I &= -\frac{R^2}{2\left(R^2 - x_c^2\right)^{3/2}} \end{aligned} \tag{5.15}$$

Para medir el error existente entre ambas expresiones, P_E y P_I, tomemos el siguiente ejemplo (figura 5.24). Sea una circunferencia que pasa por el origen de coordenadas (centro geométrico del pixel), con una pendiente

[6] Consideremos el caso $0 \le y'(0) \le 1$ e $y(0) > y_c$.

entre 0 y 1. En las gráficas se muestra el error cometido en el cálculo de los coeficientes a, b, y c para distintos valores de radio y pendiente (medida en la vertical central). Como puede apreciarse, a menor radio de curvatura mayor es el error. También puede observarse cómo el error es prácticamente nulo cerca de la pendiente horizontal, y se hace mayor conforme la pendiente tiende a 45 grados. Esto es así debido al hecho que la parábola y la circunferencia difieren más entre sí a medida que nos alejamos del eje central (como se aprecia en las figuras 5.23c y d).

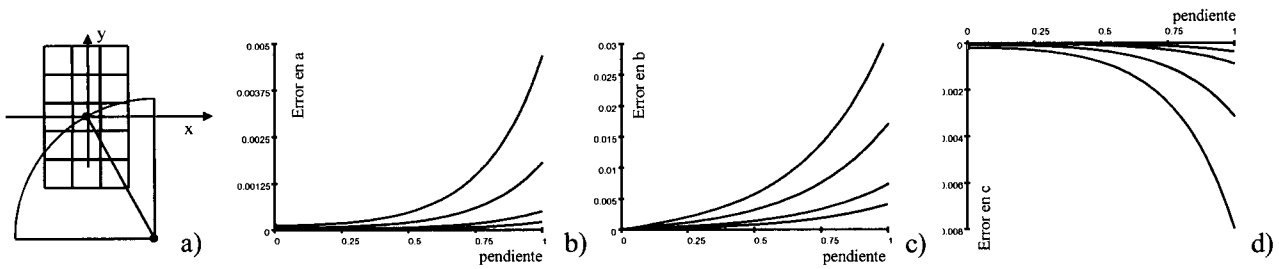


Figura 5.24: a) Circunferencia de radio R que pasa por el origen con pendiente entre 0 y 1. b,c,d) Errores cometidos en el cálculo de los coeficientes a, b y c en función de la pendiente para varios radios: 15 (verde), 20 (rojo), 30 (azul) y 40 (negro).

Pongamos como ejemplo concreto una circunferencia de radio 15 que toca el origen de coordenadas con pendiente 1, cuya expresión es

$$\left(x - \frac{15}{2}\sqrt{2}\right)^2 + \left(y + \frac{15}{2}\sqrt{2}\right)^2 = 225$$

La parábola ideal P_I que comparte posición, pendiente y curvatura en $x = 0$, viene dada por la expresión

$$P_I(x) = x - \frac{1}{15}\sqrt{2}x^2 \approx x - 0.094x^2$$

La parábola P_E estimada por nuestro método a partir de la imagen de la circunferencia viene dada aproximadamente por

$$P_E(x) = 0.0047 + 1.032x - 0.102x^2$$

La pregunta es: ¿cómo podríamos obtener P_I a partir de P_E? Pues para ello, planteamos el esquema numérico que se propone en la siguiente sección.

Detección de circunferencias

Sea una imagen ideal de un arco de circunferencia de radio R y centro en el punto (x_c, y_c), que pasa por la vecindad de un cierto pixel. Lo que pretendemos es deducir el valor de dichas variables (radio y centro) a partir de la parábola P_0 obtenida por nuestro método detector al procesar la imagen.

Si dichas variables fuesen conocidas a priori, sabemos que la expresión 5.13 nos da la parábola P_E estimada por nuestro detector, donde U_x es la función primitiva de la circunferencia (ecuación 5.14). Por otro lado también sabemos cuál es la parábola P_I ideal que nos gustaría obtener, que es aquélla que comparte posición, pendiente y curvatura con la circunferencia en el punto $x = 0$, y que viene dada por la expresión 5.15.

Supongamos que ambas parábolas vienen dadas por los coeficientes

$$\begin{aligned} P_E(x) &= a_E + b_E x + c_E x^2 \\ P_I(x) &= a_I + b_I x + c_I x^2 \end{aligned}$$

Haciendo las restas de cada pareja de coeficientes

$$\begin{aligned} \varepsilon_a &= a_E - a_I \\ \varepsilon_b &= b_E - b_I \\ \varepsilon_c &= c_E - c_I \end{aligned}$$

obtenemos tres factores de corrección que, restados a los coeficientes obtenidos por nuestro método, nos darían los coeficientes de la parábola buscada. Estos factores serán siempre muy pequeños (en el ejemplo anterior de radio 15 el mayor es inferior a 0.04).

Pero el problema es que la circunferencia exacta es desconocida a priori. Sin embargo, tenemos una buena aproximación de ella, dada por la circunferencia que comparte posición, pendiente y curvatura en $x = 0$ con la parábola obtenida por nuestro método, la cual se obtiene con las expresiones 5.10 y 5.11.

Por tanto puede plantearse el algoritmo siguiente:

1) Dado un pixel por el que pasa una circunferencia desconocida, C, aplicamos nuestro método detector para obtener una parábola P_0, a partir de la cual se obtiene la estimación de la posición, pendiente y curvatura del contorno.

2) Con dichos valores deducimos el centro y radio de la circunferencia C_0.

3) Como sabemos que C_0 es una buena aproximación de C, podemos calcular los factores de corrección ε para corregir P_0 y obtener una nueva parábola corregida, P_1.

4) A partir de P_1 puede estimarse la circunferencia C_1 que será una mejor aproximación de C.

De hecho, este esquema puede repetirse sucesivamente hasta converger. Es decir, en cada iteración n, partimos de una circunferencia C_n con la cual calculamos los factores de corrección ε para corregir la parábola inicial P_0 y obtener P_{n+1} de la cual deducimos una nueva circunferencia C_{n+1}. El algoritmo quedaría como sigue

Funcion EstimarCircunferencia (P, C)

```
P0 = P
Repetir hasta convergencia
   Deducir la circunferencia C a partir de P (ec.10 y 11)
   Calcular la parabola estimada Pe (ec.13)
   P += P0 - Pe
Fin Repetir
```

Se ha hecho una simplificación en el algoritmo, ya que en cada iteración, P_I coincide con la parábola de la iteración anterior. El algoritmo parte de la parábola obtenida por el método detector, P, y en cada iteración se va obteniendo una nueva circunferencia C, que finalmente resulta ser la que buscamos. En la tabla 5.4 se muestra la traza al ejemplo anterior de radio 15. Podemos ver cómo en muy pocas iteraciones se logra la convergencia a los valores buscados.

Adición de un pixel a la ventana

Para lograr la precisión total en contornos circulares ideales, nos falta aún un último detalle, consistente en arreglar el problema que se muestra en la figura 5.25a. Supongamos que un arco de circunferencia cruza el pixel etiquetado como rojo con pendiente menor que 1, con lo cual se generará en cada iteración una ventana vertical $9x3$ centrada en dicho pixel. Supongamos también que dicho arco cruza el pixel etiquetado como verde con pendiente mayor que 1, con lo cual se generará una ventana horizontal $3x9$.

Tal y como hemos propuesto el algoritmo anterior, todos aquellos pixels que al final de cada iteración no hayan sido incluidos en ninguna de las ventanas generadas sintéticamente, se les supone que no están cerca de

	Circunferencia			Par. estimada			Parabola final		
Iteraciones	R	x_c	y_c	a	b	c	a	b	c
0							+0.0047	1.0320	−0.1023
1	14.504	10.416	−10.088	0.0108	1.0704	−0.1127	−0.0014	0.9936	−0.0919
2	15.244	10.744	−10.815	0.0029	1.0239	−0.0993	+0.0004	1.0018	−0.0949
3	14.939	10.573	−10.554	0.0052	1.0343	−0.1031	−0.0001	0.9995	−0.0941
4	15.017	10.616	−10.621	0.0046	1.0314	−0.1021	+0.0000	1.0001	−0.0943
5	14.995	10.604	−10.603	0.0047	1.0322	−0.1024	+0.0000	1.0000	−0.0943
6	15.001	10.607	−10.608	0.0047	1.0320	−0.1023	+0.0000	1.0000	−0.0943

Tabla 5.4: Iteraciones del esquema numérico para encontrar la circunferencia

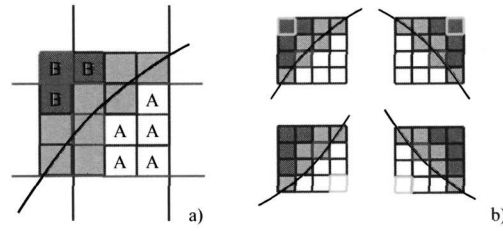


Figura 5.25: a) Si en el pixel rojo la pendiente es menor que 1, y en el pixel verde es mayor que 1, el algoritmo producirá un error en el valor del pixel amarillo. b) Las cuatro situaciones posibles para una ventanita vertical

ningún contorno, por lo que mantendrán el valor obtenido tras el suavizado. Sin embargo, en la figura puede verse que el pixel de intensidad B etiquetado como amarillo no pertenece a ninguna ventana, y sin embargo, al suavizarlo con una máscara H_1, su valor será ligeramente diferente de B, debido a la pequeña porción de contorno que toca a su vecino derecho inferior. Esto producirá que conforme avancen las iteraciones, el pixel irá perdiendo su valor original, lo cual acabará afectando a la forma del contorno.

Para solventar este problema, primero hay que detectar en qué pixel puede suceder este problema. Si consideramos $(0,0)$ las coordenadas del pixel central, y suponemos un caso vertical (pendiente b menor que 1 en valor absoluto), se producen cuatro situaciones posibles, según los signos de los coeficientes de la parábola b y c. En todos ellos, las coordenadas del pixel problemático vienen dadas por la expresión $(2s_1, s_2)$, donde s_1 es el signo del producto bc, y s_2 es el signo de c.

Una vez conozcamos la posición de dicho pixel, la idea será marcar ese pixel para que, en caso que al procesar todos los pixel borde no haya sido incluido en ninguna de las ventanas sintéticas, se le asignará el valor B estimado para la ventana. La solución que se propone es que en cada pixel borde se genere una ventana modificada $9x3+1$ (de 28 pixels, para incluir el pixel del problema), y que el peso de este último pixel dentro de la ventana sea ínfimo. Con ello conseguimos dos objetivos: en primer lugar, si dicho pixel cae en el interior de otra ventana con un peso normal, primará el valor obtenido en esta otra ventana. Y en segundo lugar, si el pixel no aparece en ninguna otra ventana, tendrá el valor estimado en la ventana inicial, y no será suavizado.

Algoritmo final

El algoritmo optimizado final, ya incluyendo la detección de circunferencias, queda como sigue (sólo se incluye la función central del algoritmo):

Funcion ActualizarImagenesOptimizado (C, I, i, j, orientacion)

	Err. módulo		Err. dirección		Err. posición		Radio de curvatura		
Ruido	Media	Max.	Media	Max.	Media	Max.	Media	Mín.	Máx.
0	0.00	0.00	0.00	0.00	0.00	0.00	20.0	20.0	20.0
20	0.48	0.66	0.76	2.00	10.8	25.2	20.0	17.2	22.9
40	0.74	0.94	1.54	4.25	26.8	67.8	19.9	15.1	30.5
60	1.06	1.30	1.92	5.31	30.4	85.2	19.8	15.2	35.4

Tabla 5.5: Errores cometidos por nuestro método tras 1000 iteraciones para un circulo de radio 20 y distintos niveles de ruido

```
Segun orientacion vertical / horizontal:
Calcular parabola P segun Lema 4.5 / Lema 4.6
EstimarCircunferencia (P,C)
Para todos los pixels (m,n) pertenecientes a una ventana 9x3+1 centrada en (i,j)
   Intensidad = Calcular intensidad del pixel (m,n) a partir de la circunferencia C
   Segun el valor de m:
      i:        I(m,n) += Peso1 * Intensidad
                C(m,n) += Peso1
      i+1, i-1: I(m,n) += Peso2 * Intensidad
                C(m,n) += Peso2
      i+2, i-2: I(m,n) += Intensidad
                C(m,n) ++
   Fin Segun
Fin Para
```

Como vemos, el peso de cada columna en la ventana es diferente, para dar mayor importancia a la columna central, y un peso muy pequeño a las columnas más exteriores. Por tanto, debe cumplirse que peso1 > peso2 > 1. En las pruebas se ha tomado peso1=1000 y peso2=10.

En la tabla 5.5 podemos ver el error cometido en los parámetros del contorno de un círculo de radio 20 para distintos niveles de ruido. Cuando no hay ruido y la imagen es ideal, ésta permanece inalterable. Cuando el ruido aumenta, también lo hace el error de cada parámetro, aunque sin ser demasiado elevados. Por ejemplo, para un ruido de magnitud 60, los errores medios a lo largo del perímetro son aproximadamente 1 unidad para el salto de intensidad (en la imagen el salto es de 100 unidades), 2 grados para la orientación, un 30% del tamaño del pixel para la posición y 0.2 unidades para la estimación del radio de curvatura.

Visualmente, el resultado es bastante bueno, como puede apreciarse en la figura 5.26. En ella puede verse como el contorno del círculo es detectado en todos los pixels del perímetro.

Cuando el ruido es muy alto hay que mencionar varios detalles. En primer lugar, el esquema numérico para estimar la circunferencia exacta no siempre va a converger en zonas con alto ruido. En estos casos, lo mejor es seguir con los parámetros estimados por el método inicial. Una alternativa válida también es aplicar inicialmente algunas pocas iteraciones usando el método lineal para disminuir la magnitud del ruido, ya que éste método es más robusto que el cuadrático. Hay que tener en cuenta que el valor de la curvatura es extremadamente sensible. Por lo tanto, funcionará mejor si el ruido inicial no es demasiado alto.

Otro problema diferente ocurre cuando el radio de curvatura es muy pequeño. Recordemos que la condición de nuestro detector de bordes era que, en cada ventana centrada en un pixel borde sobre la que se realiza la estimación del contorno, éste debe cruzar siempre de izquierda derecha (para ventanas verticales). Esto sólo puede garantizarse para radios superiores a un cierto umbral (ecuación 4.14). Por tanto, cuando la curvatura detectada sea mayor que dicho umbral, supondremos que es debido al ruido, y generaremos una curvatura igual a la umbral.

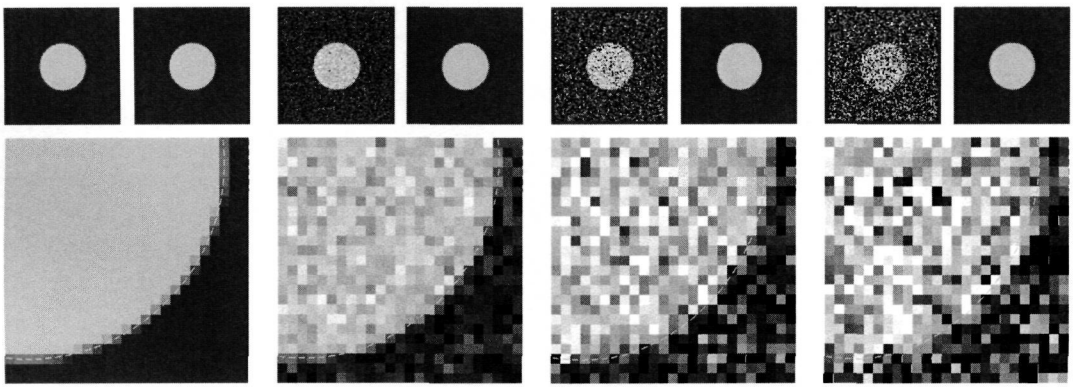


Figura 5.26: Resultado de aplicar 1000 iteraciones de nuestro método a un círculo de radio 20 con niveles de ruido 0, 20, 40 y 60. Fila inferior: ampliación de la imagen, mostrando los contornos detectados en la imagen restaurada, pero visualizándolos sobre los pixels de la imagen inicial.

Lo anterior significa que los contornos con curvatura alta se irán perdiendo durante el proceso iterativo. Este resultado puede apreciarse en la figura 5.27, donde se muestran diversas iteraciones del método lineal y del cuadrático aplicados a la imagen de un cuadrado. El método lineal degenera hasta hacer desaparecer por completo el objeto. Por el contrario, el método cuadrático sólamente deforma los puntos con curvatura alta (las esquinas del cuadrado) hasta llevarlos a la curvatura umbral, dejando intacto el resto del perímetro. En el próximo capítulo propondremos una solución para tratar de no perder estos puntos de alta curvatura

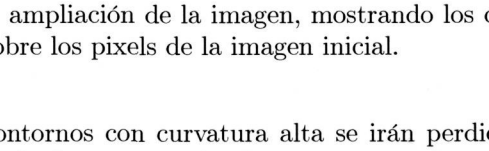


Figura 5.27: Fila superior de izquierda a derecha: método de restauración lineal aplicado a un cuadrado tras 0, 50, 100, 150 y 200 iteraciones. Fila inferior: ídem con el método cuadrático.

5.4 Ejemplos con imágenes reales

El método planteado hasta el momento funciona muy bien en las imágenes sintéticas anteriores, pero aún contiene algunas limitaciones que habrá que corregir para poder aplicarlo a imágenes reales con resultados óptimos. Si bien es cierto que el método mantiene las características ya comentadas, como disminución efectiva del ruido, o autoenfoque de contornos, en una imagen real con mucho detalle ocurre que los contornos se van deformando poco a poco en cada iteración.

Por ejemplo, en la figura 5.28 podemos ver el resultado tras veinte iteraciones con distintos valores de umbral para el gradiente. Si el valor de umbral es demasiado alto, puede que haya contornos que no superen dicho valor, con lo cual se van difuminando en cada iteración, como puede apreciarse en la parte superior del sombrero (figura d). Por otro lado, si bajamos el umbral, la imagen adquiere un aspecto pixelado, debido a que sobre un

mismo pixel caen muchas ventanas a la vez, con lo cual se entremezcla información de varios contornos sobre un mismo pixel (figura b). En cualquiera de los casos, y atendiendo a la zona de la boca, vemos que se pierde bastante detalle, volviéndose una imagen peor definida.

En la imagen del popeye puede verse esta pérdida de detalle mucho más claro (ver figura 5.29). Analizando cada una de las imágenes, vemos que la forma de los botones se ha perdido completamente, así como el detalle de la cara y el de las barras de las persianas. También el detalle de la apertura de la puerta se ha visto bastante deformado.

En el próximo capítulo analizaremos las causas de estos problemas, y propondremos soluciones para todas ellas.

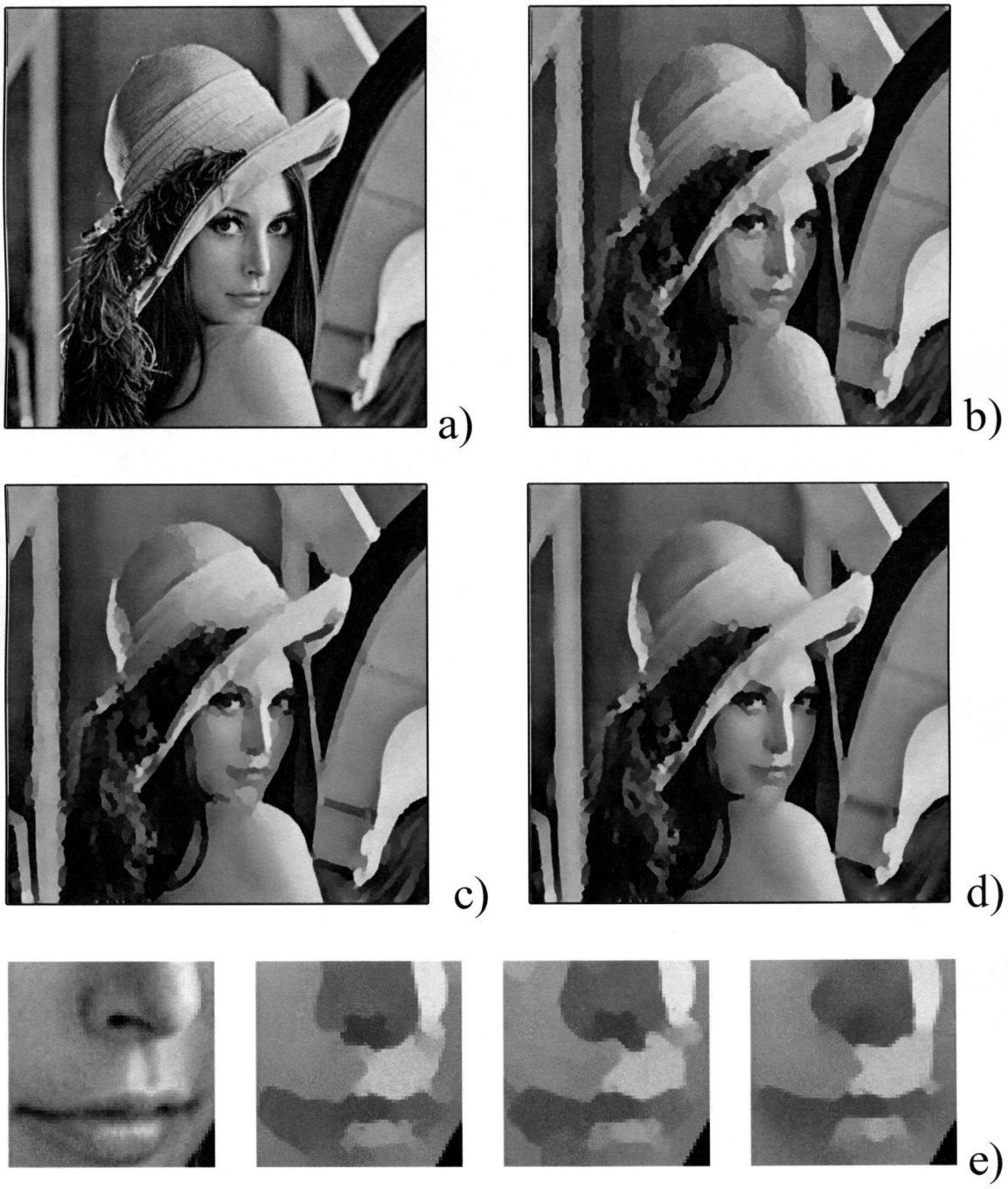


Figura 5.28: a) imagen original; b,c,d) resultado de aplicar 20 iteraciones con umbral 2, 10 y 20 respectivamente; e) detalle de las 4 imágenes

Figura 5.29: Resultado de aplicar 10 iteraciones de nuestro método de restauración

Capítulo 6

Restauración avanzada de imágenes

- Aún faltan un par de cosas por arreglar: bordes muy cercanos entre sí, y curvaturas altas.
- Vamos a por ello.....

JUNIO 2002

En este último capítulo del bloque de imágenes 2D propondremos dos nuevos algoritmos de restauración, ligeramente más complejos que los del capítulo anterior, para solventar los problemas comentados. Finalmente veremos ejemplos con imágenes reales, haciendo mayor hincapié en imágenes angiográficas, como la de la figura 6.1.

El interés en este tipo de imágenes es detectar con la mayor precisión los contornos de las venas[1], de cara a dar una información muy precisa como ayuda al diagnóstico por parte del especialista médico. Si aplicamos el proceso anterior, vemos que al utilizar nuestro método de restauración se pierde la información de los contornos. Veamos a continuación las causas de dichos problema y cómo resolverlas.

6.1 Método propuesto de restauración con límites variables

El problema fundamental del método anterior cuando es aplicado a imágenes reales es debido a una suposición inicial de nuestro método detector, que es la siguiente: en el interior de la ventana $9x3$ centrada en un pixel borde existe un único contorno. Esto puede verse en la figura 6.2: en la fila superior se muestra un contorno aislado, su imagen digitalizada, y la misma imagen después de suavizar. Para cada pixel borde, el detector trabaja con una vecindad $9x3$ de la imagen suavizada para estimar los parámetros del contorno, y para ello parte de que las zonas superior e inferior de la ventana (para un caso vertical) se corresponden con los valores de intensidad a cada lado del contorno.

Pero ¿qué ocurre cuando existe un segundo contorno cercano al que estamos tratando de estimar? En el ejemplo de la fila inferior de la misma figura se aprecia cómo la presencia de este segundo contorno produce que, tras el proceso de suavizado, los valores de la zona superior de la ventana no coincidan con la intensidad B. Esta variación hará que al aplicar el método detector, los valores estimados para el contorno sean erróneos.

Este problema ya ocurría en los ejemplos con imágenes reales que hemos usado en capítulos previos. Pero en estos casos, salvo que los contornos estuvieran demasiado cerca el uno del otro, producían errores no demasiado

[1] El término correcto sería "vasos sanguíneos", los cuales pueden ser tanto arterias como venas.

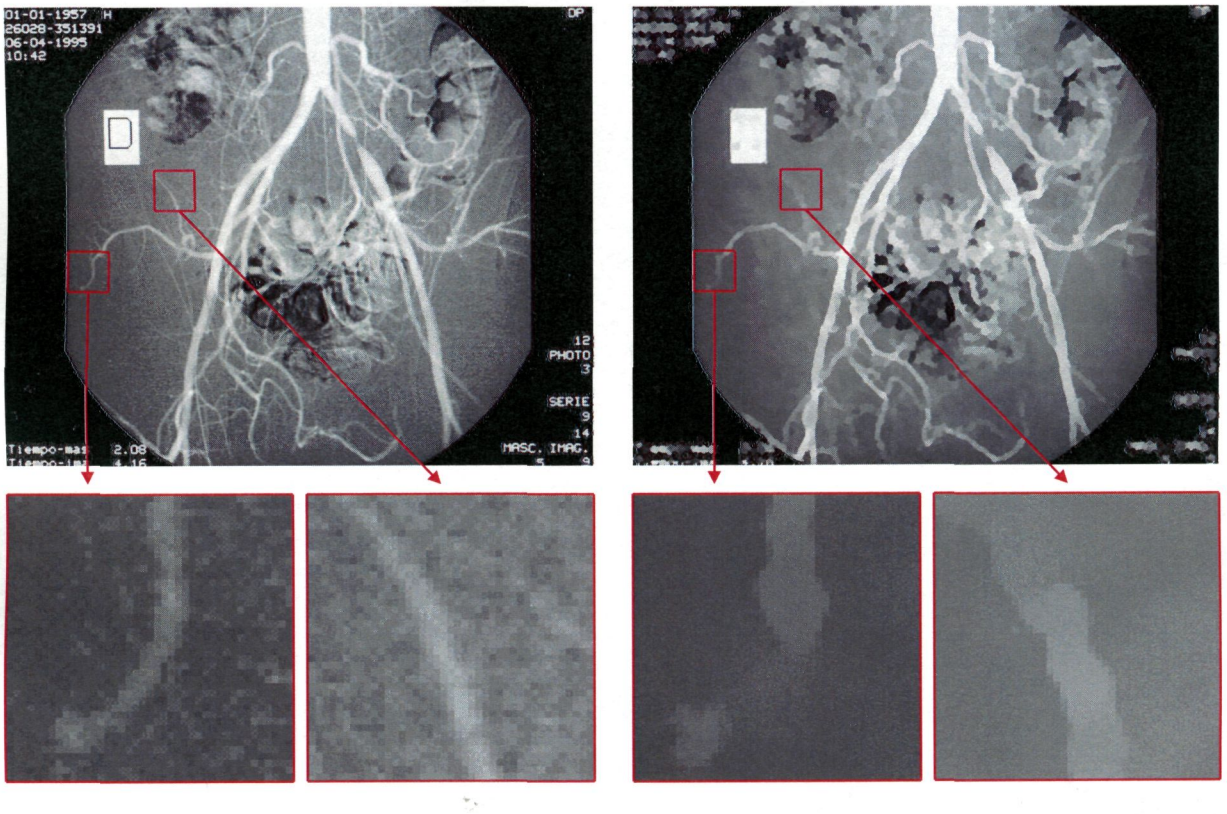


Figura 6.1: Imagen angiográfica y su restauración con el método propuesto.

graves. Sin embargo, como ahora estamos trabajando con un esquema iterativo, el problema empieza a ser importante.

En realidad, el problema viene de usar una ventana tan larga (9 pixel de alto en la imagen suavizada, lo cual representa un área de influencia de 11 pixels de alto en la imagen original) para estimar el contorno. Esto era así para garantizar en todos los casos que el contorno cruza la ventana de izquierda a derecha. Sin embargo, en la mayoría de los casos no haría falta irse tan lejos. Y en eso consiste la solución propuesta.

6.1.1 Algoritmo básico

Recordando del capítulo anterior, para el caso en que la pendiente del contorno m estaba entre -1 y 1, los parámetros del contorno se estimaban a partir de las sumas de las tres columnas centrales en la imagen suavizada G (lema 4.5), cuyas expresiones eran

$$\begin{aligned} S_L &= \sum_{k=-3+m}^{3+m} G_{i-1,j+k} \\ S_M &= \sum_{k=-3}^{3} G_{i,j+k} \end{aligned} \tag{6.1}$$

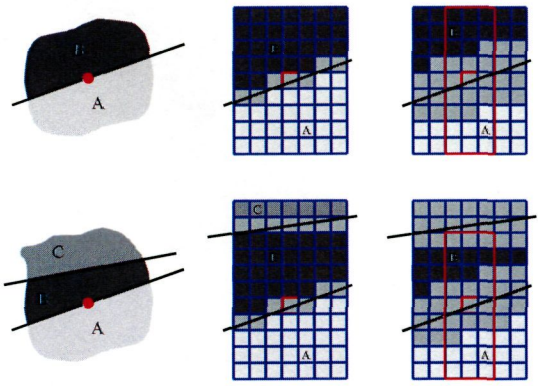


Figura 6.2: Fila superior, de izquierda a derecha: un contorno aislado, su imagen digitalizada y la imagen suavizada donde trabaja el detector. Fila inferior: dos contornos cercanos.

$$S_R = \sum_{k=-3-m}^{3-m} G_{i+1,j+k}$$

y los valores de intensidad a ambos lados del contorno salían de las expresiones

$$\begin{aligned} B &= \frac{G_{i-m,j-3} + G_{i-m,j-4} + G_{i,j-4}}{3} \\ A &= \frac{G_{i,j+4} + G_{i+m,j+4} + G_{i+m,j+3}}{3} \end{aligned}$$

Como puede verse en la figura 6.3, al aplicar este lema se obtienen con exactitud los parámetros del contorno cuando éste está aislado (figura a) pero se produce un error cuando hay otro contorno cercano (figura b). En el ejemplo de esta figura, el error es debido a una mala estimación de la intensidad B.

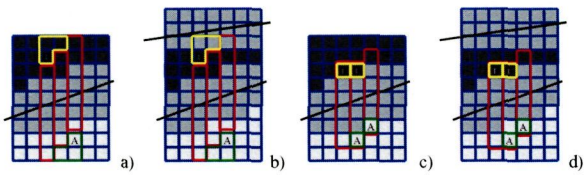


Figura 6.3: El detector propuesto estima con precisón un contorno aislado (a) pero comete error cuando hay dos contornos cercanos (b). Ajustando los límites de las columnas, el método funcionaría en ambos casos (c y d)

Para solventar este problema, se propone no forzar a realizar las sumas de las columnas entre los mismos límites siempre, sino ajustarlos en función de cada caso. Es decir, la expresión de las sumas (ecuación 6.1) quedaría como sigue

$$S_L = \sum_{k=l_1}^{l_2} G_{i-1,j+k}$$
$$S_M = \sum_{k=m_1}^{m_2} G_{i,j+k}$$
$$S_R = \sum_{k=r_1}^{r_2} G_{i+1,j+k}$$

donde los límites de las sumatorias l_1, l_2, m_1, m_2, r_1 y r_2 se calculan para cada caso. ¿Cómo se calculan? Recordando de la sección 2.1.1, la imagen de parciales alrededor de un contorno sigue un perfil similar al de la figura 6.4a, donde la llave roja muestra los pixels que se utilizan en la sumatoria. Si este número es fijo, cuando haya un segundo contorno cercano, mantener ese tamaño incluirá valores del segundo contorno en la estimación del primero. Por tanto, la solución propuesta consiste en hacer llegar la sumatoria en cada columna hasta un pixel donde estimemos que hemos alcanzado el valor de la intensidad de la zona que separaba el contorno, es decir, hasta donde el valor de la derivada alcance un mínimo, como muesta la llave azul de la figura b.

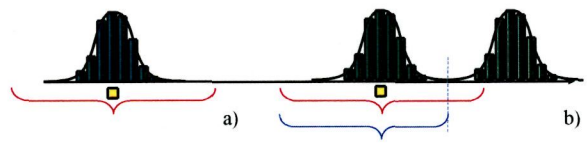


Figura 6.4: La llave roja indica los pixels usados para estimar el contorno sobre el pixel amarillo. La llave azul indica los pixels usados si ajustamos la sumatoria hasta que la derivada alcance un mínimo

Finalmente, para estimar las intensidades A y B se utilizarán los pixels extremos de las sumatorias, tal y como se muestra en la figura 6.3d, es decir

$$B = \frac{G_{i-1,j+l_1} + G_{i,j+m_1}}{2} \tag{6.2}$$
$$A = \frac{G_{i,j+m2} + G_{i+1,j+r2}}{2}$$

para los casos de pendiente positiva, y

$$B = \frac{G_{i,j+m_1} + G_{i+1,j+r_1}}{2}$$
$$A = \frac{G_{i-1,j+l2} + G_{i,j+m2}}{2}$$

para los casos de pendiente negativa.

Expresiones del método suavizado para límites variables

Sólamente nos falta encontrar las nuevas expresiones para los coeficientes a, b, c de la parábola estimada cuando los límites de la sumatoria no son fijos. Para ello, vamos a rehacer el desarrollo de la sección 4.4 pero dejando como variables dichos límites.

En primer lugar, llamemos $P_{i,s,t}$ al área interior a la columna i bajo la curva del contorno, considerando que la columna está formada por los pixels (i,s), $(i,s+1)$, ..., $(i,t-1)$, (i,t). Asumiendo que el contorno cruza la columna siempre de izquierda a derecha, su expresión será como sigue:

$$P_{i,s,t} = \int_{(i-\frac{1}{2})h}^{(i+\frac{1}{2})h} \left(a + bx + cx^2 + \left(t + \frac{1}{2} \right) h \right) dx \tag{6.3}$$

Ahora llamemos $M_{i,s,t}$ al resultado de sumar los valores de los pixels de dicha columna en la imagen original. Suponemos que las intensidades a ambos lados del contorno son A (debajo) y B (encima). Su expresión entonces será

$$M_{i,s,t} = \sum_{j=s}^{t} F_{i,j} = \frac{1}{h^2}(A-B)P_{i,s,t} + B\,(t-s+1) \tag{6.4}$$

donde F es la imagen original antes de suavizar. Finalmente, la suma de los mismos pixels pero en la imagen suavizada, G, tendrá la expresión siguiente:

$$\begin{aligned} S_{i,s,t} &= \sum_{j=s}^{t} G_{i,j} = a_{00} M_{i,s,t} + a_{01}\left(M_{i-1,s,t} + M_{i,s+1,t-1} + M_{i,s-1,t+1} + M_{i+1,s,t}\right) + \\ &\quad + a_{11}\left(M_{i-1,s-1,t+1} + M_{i-1,s+1,t-1} + M_{i+1,s+1,t-1} + M_{i+1,s-1,t+1}\right) \end{aligned} \tag{6.5}$$

De esta forma, podemos plantear el sistema de ecuaciones usando las sumas reales de las columnas izquierda, central y derecha de la imagen suavizada, y deducir las expresiones para los coeficientes de la parábola a, b, c. Considerando $(0,0)$ las coordenadas del pixel central donde queremos estimar los parámetros del contorno, el sistema queda como sigue:

$$\left.\begin{array}{c} S_L = S_{-1,l_1,l_2} \\ S_M = S_{0,m_1,m_2} \\ S_R = S_{1,r_1,r_2} \end{array}\right\}$$

cuya solución es

$$\begin{aligned} c &= \frac{S_L + S_R - 2S_M}{2h\,(A-B)} + \frac{A\,(2m_2 - l_2 - r_2) - B\,(2m_1 - l_1 - r_1)}{2h\,(A-B)} \\ b &= \frac{S_R - S_L}{2(A-B)} + \frac{A(l_2 - r_2) - B(l_1 - r_1)}{2(A-B)} \\ a &= \frac{2S_M - (1+2m_2)\,A - (1-2m_1)\,B}{2\,(A-B)} h - ch^2 \frac{1 + 24a_{01} + 48a_{11}}{12} \end{aligned} \tag{6.6}$$

donde a se ha dejado en función de c para mayor simplicidad. Podemos ver cómo el método anterior de límites fijos es un caso particular de estas expresiones, donde los valores de los límites son $l_1 = -2$, $l_2 = 4$, $m_1 = -3$, $m_2 = 3$, $r_1 = -4$ y $r_2 = 2$.

Todo lo anterior se resume en el lema siguiente, suponiendo un caso de pendiente menor que 1 en valor absoluto.

Lema 6.1 *Sea F una imagen por cuyo pixel (i,j) pasa un contorno que separa dos zonas de intensidad A (debajo del contorno) y B (encima del contorno). Dicha imagen ha sido suavizada con una máscara de suavizado H_1, tal que*

$$H_1 = \begin{pmatrix} a_{11} & a_{01} & a_{11} \\ a_{01} & a_{00} & a_{01} \\ a_{11} & a_{01} & a_{11} \end{pmatrix}$$

dando como resultado la imagen G, donde se cumple que

$$|G_y(i,j)| > |G_x(i,j)|$$

A partir de los datos de la imagen suavizada G podemos estimar los parámetros del contorno de la imagen original F (posición, pendiente, magnitud del salto de intensidad y curvatura, medidos sobre la vertical central del pixel), realizando los siguientes pasos:

a) obtenemos las sumas de las columnas izquierda, central y derecha de la siguiente manera:

$$\begin{aligned} S_L &= \sum_{k=l_1}^{l_2} G_{i-1,j+k} \\ S_M &= \sum_{k=m_1}^{m_2} G_{i,j+k} \\ S_R &= \sum_{k=r_1}^{r_2} G_{i+1,j+k} \end{aligned} \tag{6.7}$$

donde $-4 < l_1, m_1, r_1 < 0$ y $0 < l_2, m_2, r_2 < 4$ son tales que $|G_y(i-1,j+l_1)|$, $|G_y(i-1,j+l_2)|$, $|G_y(i,j+m_1)|$, $|G_y(i,j+m_2)|$, $|G_y(i+1,j+r_1)|$, y $|G_y(i+1,j+r_2)|$ son mínimos de su columna

b) obtenemos también una estimación para los valores de intensidad a ambos lados del borde

$$B = \begin{cases} \frac{G_{i-1,j+l_1}+G_{i,j+m_1}}{2} & \text{si } G_x(i,j)G_y(i,j) > 0 \\ \frac{G_{i,j+m_1}+G_{i+1,j+r_1}}{2} & \text{si } G_x(i,j)G_y(i,j) < 0 \end{cases}$$

$$A = \begin{cases} \frac{G_{i,j+m_2}+G_{i+1,j+r_2}}{2} & \text{si } G_x(i,j)G_y(i,j) > 0 \\ \frac{G_{i-1,j+l_2}+G_{i,j+m_2}}{2} & \text{si } G_x(i,j)G_y(i,j) < 0 \end{cases}$$

c) a continuación detectamos los coeficientes de la ecuación $y = a + bx + cx^2$ como sigue:

$$c = \begin{cases} 0 & \text{si buscamos contornos rectos} \\ \frac{S_L+S_R-2S_M}{2(A-B)} + \frac{A(2m_2-l_2-r_2)-B(2m_1-l_1-r_1)}{2(A-B)} & \text{si buscamos contornos de segundo grado} \end{cases}$$

$$b = \frac{S_R - S_L}{2(A-B)} + \frac{A(l_2-r_2) - B(l_1-r_1)}{2(A-B)}$$

$$a = \frac{2S_M - (1+2m_2)A - (1-2m_1)B}{2(A-B)} - \frac{1+24a_{01}+48a_{11}}{12}c$$

d) finalmente, calculamos los parámetros del contorno

$$\begin{aligned} \text{corte con la vertical central del pixel:} \quad & (0, a) \\ \text{vector normal:} \quad & N = \frac{A-B}{\sqrt{1+b^2}} \left[\begin{array}{cc} b, & -1 \end{array} \right] \\ \text{curvatura} \quad : \quad & K = \frac{2cn}{(1+b^2)^{3/2}} \end{aligned}$$

donde

$$n = \begin{cases} 1 & \text{si } G_y(i,j) > 0 \\ -1 & \text{si } G_y(i,j) < 0 \end{cases}$$

Generación de ventanas de tamaño variable Aplicando el lema anterior sobre un pixel borde ya podemos obtener los parámetros precisos del contorno con mayor rigor que de la forma en que se hacía en el capítulo previo. Además de los coeficientes calculados para la parábola a, b, c, los límites de las sumatorias son también un dato importante de cara a la generación de la ventana sintética sobre la nueva imagen que se está generando. Ahora ya no serán siempre ventanas fijas de $9x3$, sino que solamente generaremos valores sintéticos para aquellos pixels que fueron incluidos en la sumatoria.

Ejemplos

Probemos primero con dos "venas" sintéticas, una recta y otra circular, de tan sólo 4 pixels de grosor (figura 6.5). Vemos que el método de límites fijos degenera la forma de la vena, debido al error que comentábamos antes. Sin embargo, el método de límites variables no altera la imagen, detectando con precisión todos los parámetros del contorno (posición, pendiente, curvatura y salto de intensidad). Incluso en el caso de añadir ruido a la imagen, el resultado es bastante bueno en relación al método anterior.

Si ahora retomamos la imagen angiográfica de la figura 6.1 el resultado es bastante mejor, como se ve en la figura 6.6, ya que los contornos no se deforman tanto como antes, dando lugar a una estimación bastante correcta.

En el ejemplo sintético hemos probado con contornos separados por 4 pixels de distancia. Pero en la angiografía que hemos utilizado hay algunas venas de 3 y 2 pixels de grosor. ¿Funcionará nuestro método en estos casos? Si rehacemos las imágenes sintéticas de la figura 6.5 para grosores inferiores veremos que el algoritmo falla, degenerando los contornos. A continuación vamos a analizar cuál es el problema en estos casos, y propondremos un nuevo algoritmo para solucionarlo.

6.1.2 Detección de contornos muy cercanos

Cuandos dos contornos están muy cercanos entre sí, estaremos en una situación como la que muestra la figura 6.7. Podemos ver que el valor de la intensidad B entre los dos contornos se ha perdido en la imagen suavizada. Por tanto, aunque extendamos la sumatoria hasta alcanzar un mínimo de la derivada, en ese punto el valor de la intensidad no será el correcto, y ello producirá un error.

¿Cómo podemos detectar cuándo estamos en esta situación? Atendiendo a la figura 6.4b, cuando los contornos no estaban tan próximos, puede verse que la derivada parcial prácticamente se anula, antes de comenzar a subir de nuevo. Sin embargo, cuando los contornos están muy próximos, antes de llegar a un valor casi nulo, la subida comienza con fuerza muy pronto. Es decir, la condición para detectar este caso podría ser mirar el valor de la parcial uno o dos pixels más allá de aquél donde se hace mínima, y ver si supera un cierto umbral.

Por ejemplo, cojamos la figura 6.8a. En ella se muestra una vena sintética de 3 pixels de grosor, con intensidad 200 sobre un fondo 100 (podemos ver tanto los niveles de gris como los valores numéricos exactos

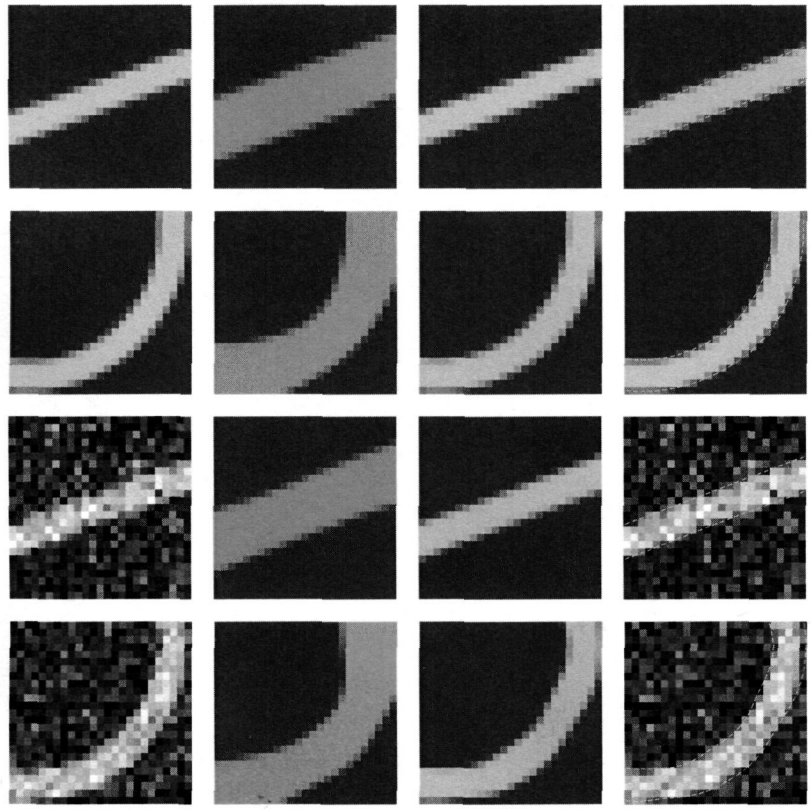


Figura 6.5: Venas de 4 pixels de grosor. Filas superiores, de izquierda a derecha: imagen original, restaurada con límites fijos, restaurada con límites variables, y contornos estimados. Filas inferiores: ídem con ruido

de cada pixel). En la figura b tenemos la imagen suavizada. Centrémonos en el pixel marcado en azul, cuyo contorno queremos estimar. Sabemos que dicho pixel es un pixel borde porque en la imagen de parciales (figura c) adquiere un valor máximo en su columna (recordemos que estamos en un caso de pendiente menor que 1).

Si aplicamos el método de límites variables haríamos las sumas de las columnas de pixels que se muestran en rojo en la figura b. Si observamos los valores de los pixels marcados en verde, que son precisamente los que deberían usarse para estimar el valor de B, vemos que efectivamente ambos tienen un valor incorrecto, debido a la influencia del contorno superior. ¿Cómo saber que se trata de una situación de contornos muy cercanos? Atendiendo a la imagen de parciales, vemos que los pixels superiores a cualquiera de los marcados en verde tienen un valor muy alto, señal de que otro contorno está próximo.

Una vez detectada esta situación, la solución para estimar el verdadero valor de B está en la imagen original. Suponiendo un caso ideal, aunque ese pixel tenga un valor diferente de B en la imagen suavizada, es exactamente B en la imagen original, ya que es un pixel que pertenece por completo al interior de la vena. Esto es así siempre y cuando el grosor no esté muy por debajo de 2 pixels, porque si el grosor es 1 pixel o menos, la intensidad del interior no podría recuperarse, ya que no existiría ningún pixel que cayera por completo en el interior de la vena. Por lo tanto, reescribiremos la expresión 6.2 de la forma siguiente:

$$B = \frac{F_{i-1,j+l_1} + F_{i,j+m_1}}{2}$$

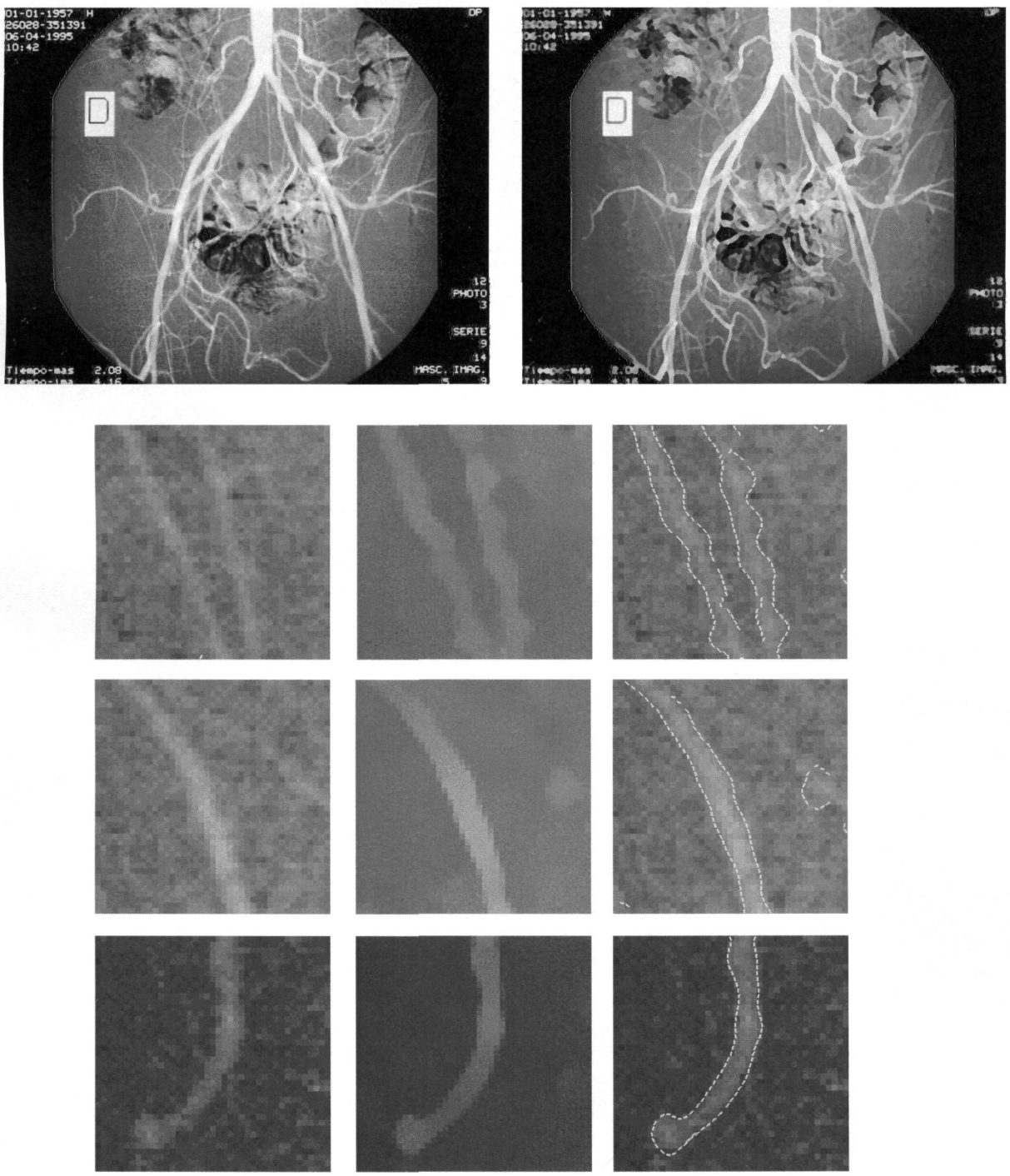


Figura 6.6: Restauración de una angiografía con el método de límites variables

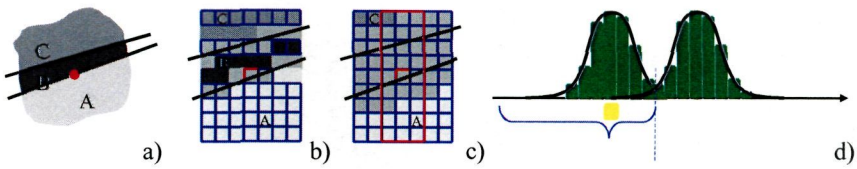


Figura 6.7: a) dos contornos muy cercanos; b) imagen digitalizada; c) imagen suavizada; d) perfil de la imagen de derivadas parciales

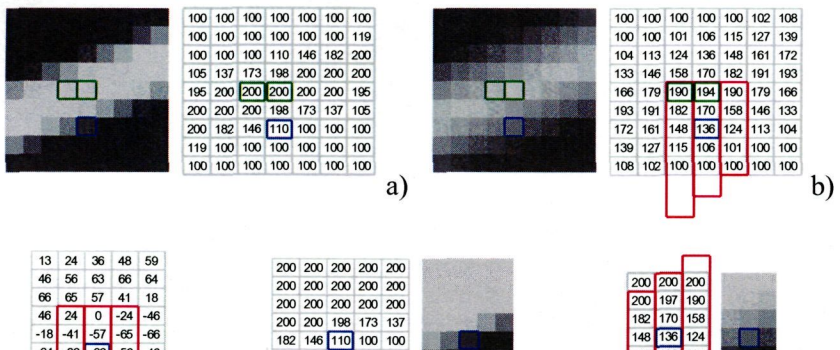


Figura 6.8:

donde F es la imagen original. Para el valor de A, en el ejemplo de la figura, no es necesario usar la imagen original, ya que por debajo se ve que no hay ningún contorno próximo. Por lo tanto, mantendremos su expresión anterior.

Generación de una subimagen intermedia Ahora que hemos estimado un valor correcto para B aparece un segundo problema: cuando apliquemos las expresiones 6.6 para obtener los coeficientes de la parábola, obtendremos un valor incorrecto, aún habiendo estimado correctamente el valor de B. Este error es debido a que dichas expresiones fueron desarrolladas a partir de la expresión 6.4 de M, que representaba el valor de la sumatoria de los pixels en la imagen original. Dicha expresión suponía que la intensidad sobre el contorno era B en todos los pixels, cosa que no ocurre en nuestro ejemplo, atendiendo a la figura 6.8a.

La solución propuesta consiste en generar una imagen alternativa F', de tamaño máximo $11x5$, centrada en el pixel borde cuyo contorno se está estimando, como la mostrada en la figura 6.8d. Dicha imagen coincidirá con los pixels de la imagen original F excepto la zona situada sobre los pixels que se usaron para estimar la intensidad B. En dicha zona, el valor será igual a B para todos los pixels. Visualmente, dicha imagen es en realidad una versión idéntica a la original pero donde el segundo contorno ha sido eliminado.

De esta forma, esta imagen F' puede ser suavizada para obtener una nueva imagen G',de tamaño máximo $9x3$, centrada en el mismo pixel, como la que se muestra en la figura 6.8e. Dicha imagen G' será sobre la que aplicaremos nuestro método de límites variables. Podemos ver en la figura cómo la zona superior izquierda de dicha subimagen ha perdido toda la influencia debida al segundo contorno, y por lo tanto, aplicar las expresiones en ella nos dará de forma exacta los parámetros deseados.

Algoritmo optimizado

En el ejemplo que hemos usado, el segundo contorno se encontraba por encima del primero. Pero cuando estimemos los parámetros del contorno para los pixels del borde superior de la vena de la figura 6.8a, la situación estará al revés, es decir, el segundo contorno estará por debajo. De hecho, podríamos encontrar una situación en otro tipo de imagen donde hubiesen contornos próximos a ambos lados simultáneamente. Por lo tanto, hay que detectar de forma separada si existe un contorno muy próximo tanto por encima como por debajo. Si no se encuentra ninguno, podemos aplicar el lema 6.1 directamente. En caso que se detecte al menos uno, habrá que generar las subimágenes F' y G', mirando qué zona es la que hay que actualizar, si la superior con el valor B, la inferior con el valor A, o ambas a la vez.

Todo ello queda resumido en la función que mostramos a continuación, que actualiza los valores de una imagen de intensidades I y otra de contadores C, para ser insertado en el algoritmo *RealzarImagen* de la página 161.

Funcion ActualizarImagenesLimitesVariables (C, I, i, j, orientacion)

```
Segun orientacion vertical / horizontal:
ContornoSuperior = ContornoInferior = False
Calcular pixels limites para la sumatoria
Si existe un segundo contorno muy cercano por arriba
   ContornoSuperior = True
   Estimar B a partir de la imagen F
Fin Si
Si existe un segundo contorno muy cercano por debajo
   ContornoInferior = True
   Estimar A a partir de la imagen F
Fin Si
Si ContornoSuperior o ContornoInferior
   Crear una subimagen F' centrada en (i,j) copiando los pixels de F
   Si ContornoSuperior actualizar B en la zona superior de F'
   Si ContornoInferior actualizar A en la zona inferior de F'
   Suavizar F' para obtener G'
   Calcular parabola P segun Lema 6.1 a partir de la imagen G'
Si No
   Calcular parabola P segun Lema 6.1 a partir de la imagen G
Fin Si
Para todos los pixels (m,n) incluidos en los limites de la sumatoria centrada en (i,j)
   Intensidad = Calcular intensidad del pixel (m,n) a partir de la circunferencia C
   Segun el valor de m:
      i:        I(m,n) += Peso1 * Intensidad
                C(m,n) += Peso1
      i+1, i-1: I(m,n) += Peso2 * Intensidad
                C(m,n) += Peso2
      i+2, i-2: I(m,n) += Intensidad
                C(m,n) ++
   Fin Segun
Fin Para
```

Ejemplos

Si repetimos las pruebas con venas sintéticas pero con grosor 2 (ver figura 6.9), la estimación de los contornos es perfecta en imágenes ideales, y bastante precisa en imágenes con ruido. La segunda columna muestra el resultado de aplicar el algoritmo básico de la sección anterior, donde se aprecia como se ensanchan los contornos.

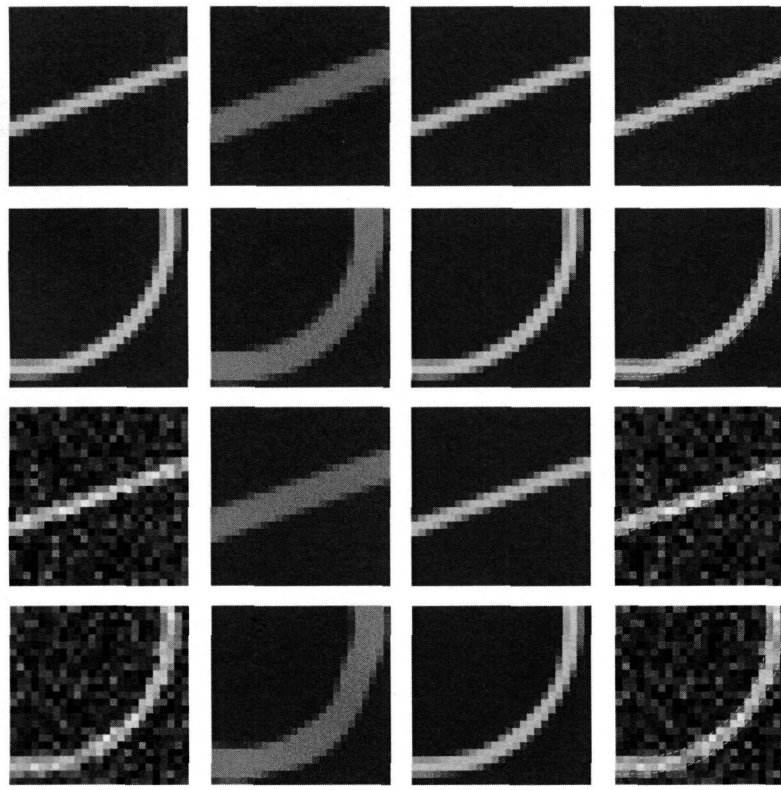


Figura 6.9: Venas de 2 pixels de grosor. Filas superiores, de izquierda a derecha: imagen original, restaurada con algoritmo básico de límites variables, restaurada con algoritmo optimizado de límites variables (con detección de contornos muy próximos), y contornos estimados. Filas inferiores: ídem con ruido

Si aplicamos el algoritmo optimizado a la imagen angiográfica de la sección anterior (ver figura 6.10), no se nota gran diferencia a simple vista. Pero cuando miramos en detalle a nivel de pixel, vemos como efectivamente, el algoritmo básico ensanchaba ligeramente las venas muy finas, debido a la influencia de ambos contornos entre sí. Sin embargo, el algoritmo optimizado corrige este efecto, ajustando mucho más los parámetros del contorno.

6.1.3 Histéresis

Ya hemos comentado anteriormente que nuestro método detector usa un parámetro umbral δ con el que indicamos el valor de salto de intensidad mínimo que debe tener un contorno para considerarlo, y por tanto generar ventanas sintéticas sobre él en cada iteración. Cualquier salto inferior a δ no será considerado un verdadero contorno sino más bien perteneciente a una zona degradada, y se irá suavizando.

Es importante recordar que dicho valor afecta a toda la imagen. Esto significa que a lo largo de la línea del contorno, el salto de intensidad puede no ser constante. Un ejemplo claro lo tenemos en la figura 6.11, donde

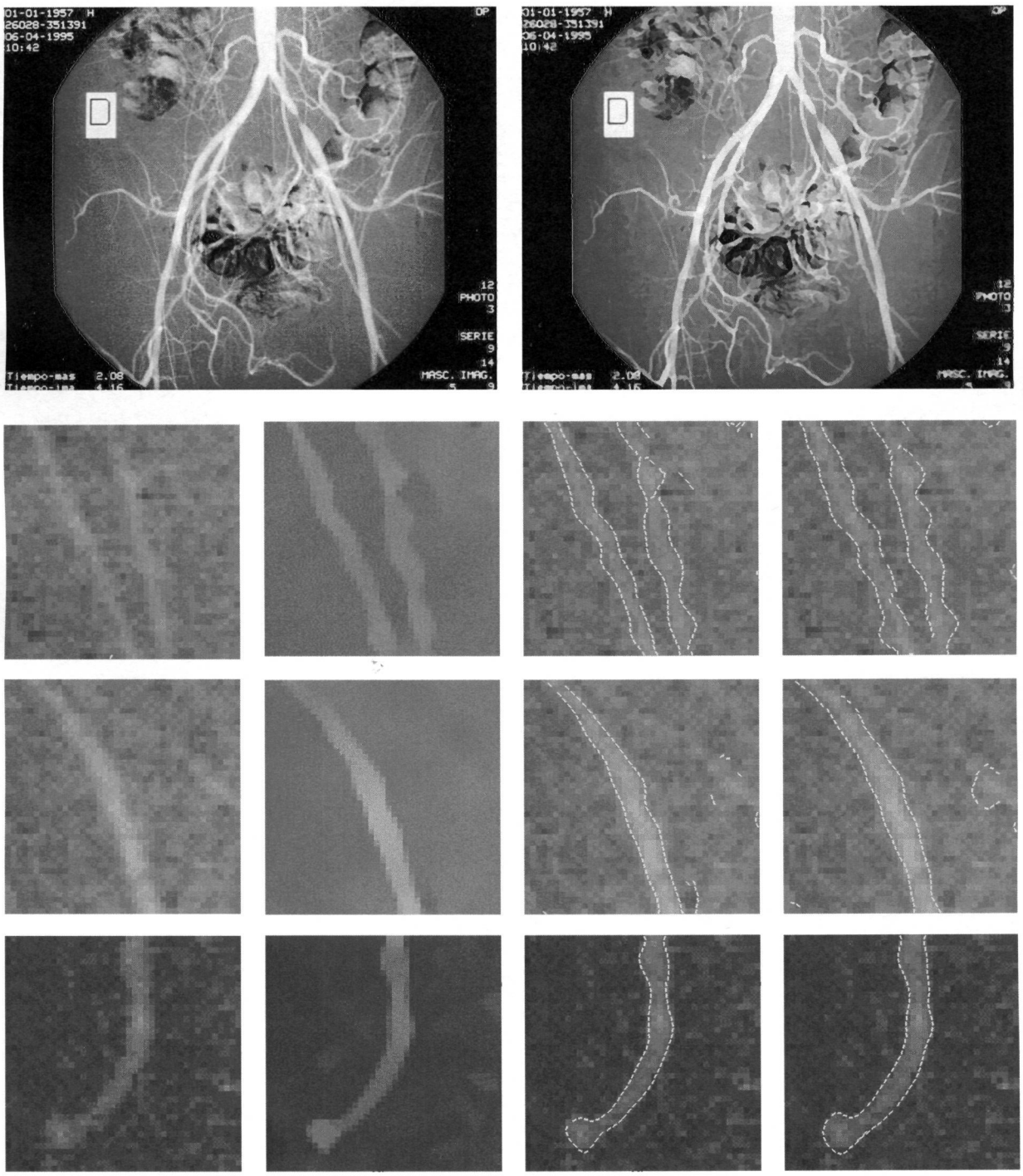


Figura 6.10: Fila superior: restauración de una angiografía con el método optimizado de límites variables. Resto de filas, de izquierda a derecha: imagen original, imagen restaurada, contornos estimados tras la restauración optimizada, y contornos estimados tras la restauración básica (copia de la figura 6.6)

en la parte superior se muestra una ampliación de la zona superior del sombrero de la chica. Puede apreciarse cómo la línea del contorno del sombrero tiene en su parte derecha un salto de intensidad de mayor magnitud que sobre la parte remarcada. Las tres primeras imágenes muestran, de izquierda a derecha, los pixels borde detectados cuya salto de intensidad es superior a 20, 15 y 10 respectivamente. Vemos que para detectar todo el contorno, es preciso bajar el umbral hasta 10, pero entonces aparecen nuevos pixels en el interior del sombrero detectados también como contorno, cuando en realidad deseamos suavizar en esa zona.

Una solución puede ser la propuesta por Canny para su famoso detector en ??, usada por muchos otros detectores posteriores, que el denominó **fase de histéresis.** Él usaba dos umbrales, uno alto, δ_H, y otro bajo, δ_L. Primero se detectaban todos los pixels borde cuya magnitud superara δ_H, y luego se iban añadiendo todos aquellos pixels vecinos a éstos cuya magnitud fuera superior a δ_L. En la imagen de la derecha de la figura 6.11 se ha tomado $\delta_H = 20$ y $\delta_L = 10$, y se han pintado de rojo los pixels borde que superan δ_H, de azul los pixels borde que superan δ_L y son vecinos de los rojos, y de verde los que también superan δ_L pero no son vecinos de los rojos. Con estos valores, estaríamos detectando toda la línea del contorno, y suavizando en el interior. De esta forma, podemos detectar contornos enteros aunque algunas zonas suyas tengan un salto de intensidad muy bajo, sin tener por ello que añadir nuevos pixels borde aislados en la imagen.

La condición de pixel vecino puede ser simplemente que esté en una posición adjunta, o refinarla aún más en función de si las orientaciones de los contornos en ambos pixels son similares. Así evitaríamos posibles bifurcaciones o pixels de ruido o textura muy cerca del contorno.

Esta fase de histéresis es opcional en cualquier caso, e iría dentro del código al comienzo de cada iteración. El pseudo código para insertar en la función *RealzarImagen* de la página 5.2.1 sería el siguiente:

```
Calcular las parciales de la imagen suavizada Gx, Gy
Marcar todos los pixels borde con magnitud superior a DeltaH
Adjuntar a partir de estos los pixels borde con magnitud superior a DeltaL
Para cada pixel borde (i,j)
   ActualizarImagenesLimitesVariables (C, I, i, j, orientacion)
```

En la misma figura podemos ver el resultado de aplicar el método de restauración optimizado con histéresis, así como el detalle de algunas zonas de la imagen. Puede comprobarse cómo el resultado es bastante mejor que el que se mostraba al final del capítulo anterior.

Por último, hemos aplicado el método de restauración a la imagen del popeye. Comparando con las obtenidas en el capítulo anterior, vemos que la restauración no ha deformado la imagen, y que los contornos se han estimado de forma bastante precisa.

6.2 Método propuesto de restauración para altas curvaturas

El último método que vamos a proponer concierne al problema de los bordes con una curvatura demasiado alta. Recordemos que todos nuestros métodos se basaban en que la línea del contorno (para pendientes menor que 1 en valor absoluto) debía cortar las ventanas siempre de izquierda a derecha. Cuando no es así, se comete un error que puede ser de consideración a medida que avancen las iteraciones. Por ejemplo, en la figura 6.12 podemos observar que los botones de la camisa del popeye, que tienen un radio aproximado de 4 pixels, adquieren una forma romboide tras varias iteraciones, y el tornillo de la puerta corredera en el límite derecho de la imagen, cuyo radio es del orden de 3 pixels, también se ve deformado. En la figura 6.13 se muestran dos ejemplos sintéticos con círculos de radio 4 y 3 pixels, donde puede apreciarse la deformación producida tras varias iteraciones.

En cualquier caso, en lo que a las imágenes angiográficas se refiere, ya que ellas son nuestro principal interés, es difícil que una vena tenga un radio de curvatura menor que 10 pixels. Por lo tanto podríamos considerar que no es necesario ser tan precisos en zonas de tan alta curvatura. Sin embargo, y ya pensando en la extrapolación de todo nuestro trabajo a las imágenes 3D, ¿qué ocurrirá cuando generemos una vena sintética en 3D de 2 ó 3 voxels de grosor, al igual que hicimos en el caso 2D? En el espacio tridimensional aparecen dos curvaturas

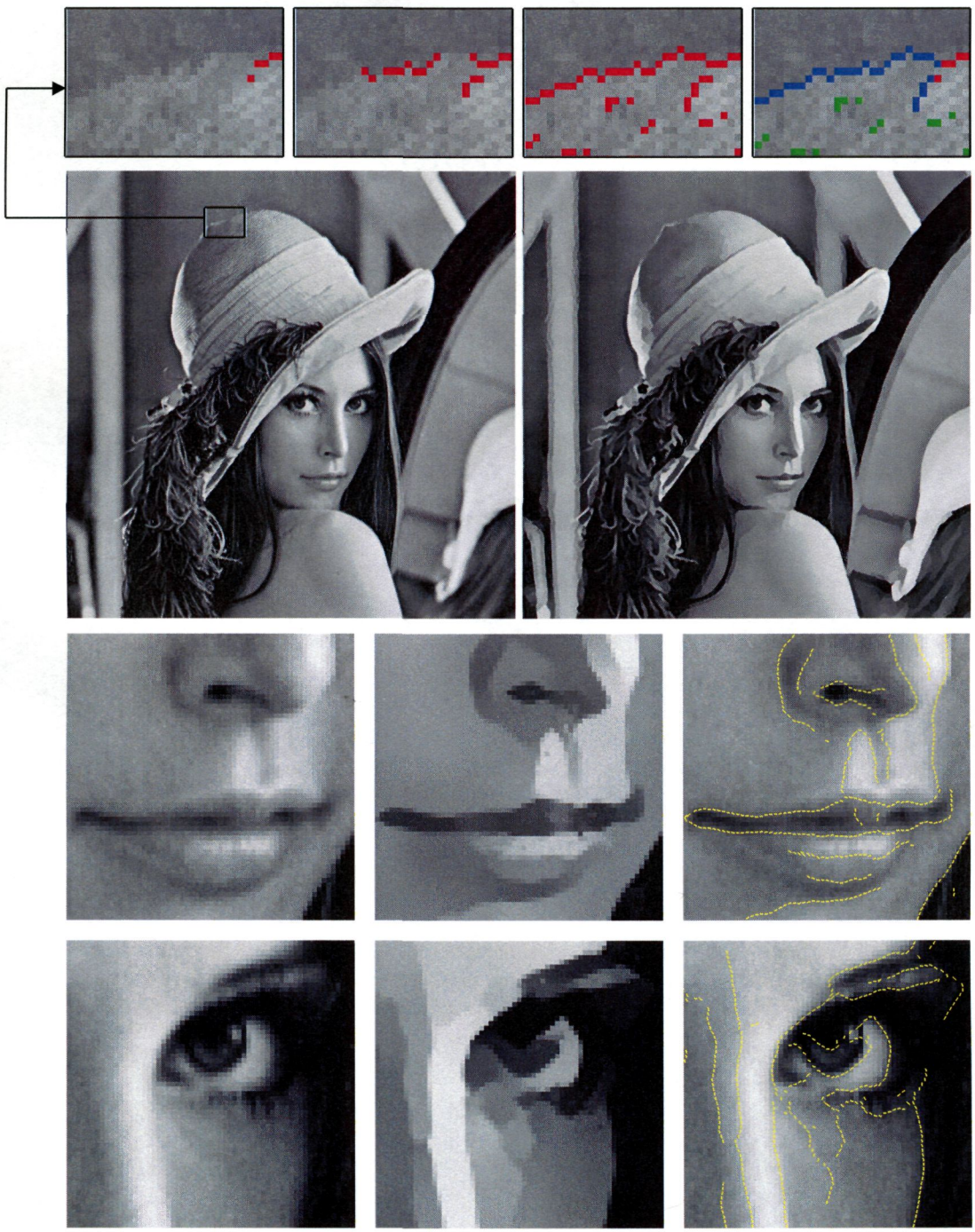


Figura 6.11: Fila superior, de izquierda a derecha: restauración sin histéresis con $\delta = 20$, 15 y 10, y restauración con histéresis con $\delta_H = 20$ y $\delta_L = 10$. Fila central: restauración optimizada con histéresis. Filas inferiores, de izquierda a derecha: imagen original, imagen restaurada, y contornos estimados.

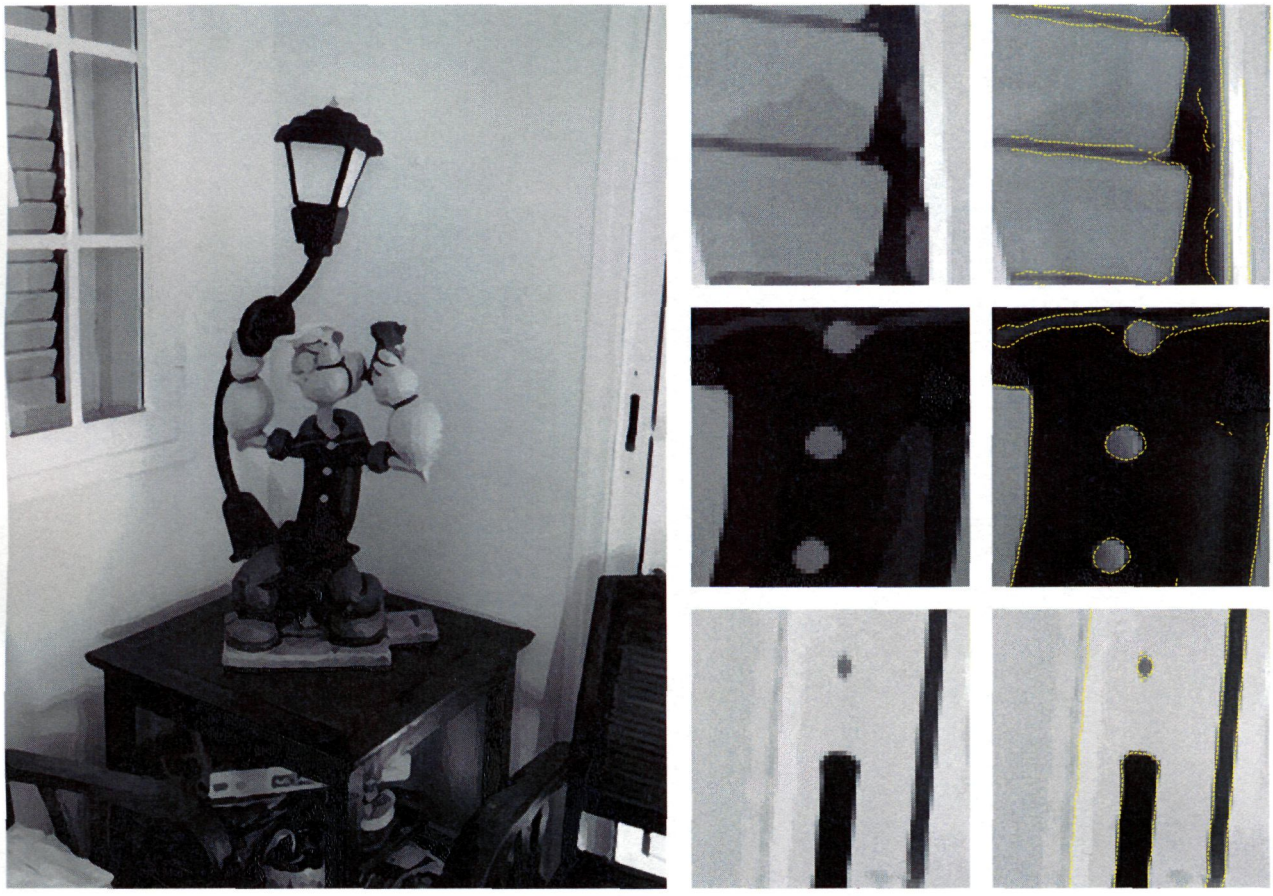


Figura 6.12: Restauración optimizada de la imagen del popeye.

diferentes. La que mide el cambio de orientación a lo largo de la vena, la cual seguramente no será demasiado alta, y la que mide la variación en la dirección perpendicular, es decir, sobre el plano transversal de la vena. Si queremos poder detectar con precisión venas de 2 ó 3 voxels de grosor, debemos detectar radios de curvatura del orden de 1 ó 2 voxels. Por lo tanto, en esta sección trataremos de resolver este último problema.

La curva no cruza de izquierda a derecha La expresión que medía el área interior a una columna de pixels bajo la curva del contorno venía dada por la ecuación 6.3. Esto erá solamente válido cuando la curva cortaba la columna de izquierda a derecha. Si no es así, los límites de la integral serían diferentes. Tomemos el ejemplo de la figura 6.14, y llamemos $(0,0)$ al pixel central marcado en rojo. Para calcular el área bajo la curva interior a la columna -2, la expresión sería

$$P_{-2,-4,4} = \int_{x_1}^{(i+\frac{1}{2})h} \left(a + bx + cx^2 + \frac{9}{2}h \right) dx$$

donde x_1 indica el corte de la curva con la base de la columna. Eso implicaría que la expresión saldría no lineal, con lo cual el sistema final de ecuaciones sería muy complejo de resolver. Por lo tanto, parece claro que

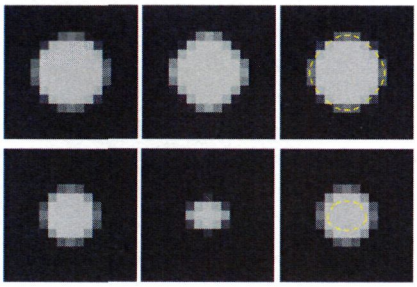


Figura 6.13: De izquierda a derecha: imagen de un círculo de radio 4 (arriba) y 3 (abajo), imagen restaurada, y contornos estimados.

analíticamente no vamos a poder obtener los parámetros de la curva, y habrá que recurrir a algún esquema numérico que nos permita encontrar con precisión el arco de circunferencia buscado.

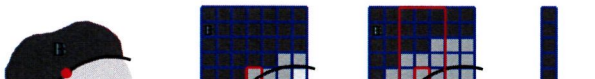


Figura 6.14: a) contorno circular; b) imagen digitalizada; c) imagen suavizada; d) área interior a la columna -2 bajo el contorno en la imagen suavizada

Por otro lado, cuando queremos detectar curvaturas altas, debemos usar ventanas centradas en el pixel del menor tamaño posible, ya que cuanto más grande sea la ventana, más promediada será la información que obtengamos. Por ejemplo, en la figura 6.15 tenemos un contorno cuyas características sobre el punto marcado en rojo queremos estimar. Aproximadamente el radio de curvatura en dicho punto es de 2 pixels. Si seguimos usando ventanas de $9x3$ (cuya influencia recordemos que se extiende a un área $11x5$ en la imagen original) obtendremos resultados diferentes si en lugar de la curva tuviésemos en la imagen un círculo completo del mismo radio, ya que las intensidades de los pixels, sobre todo en las zonas de la ventana más alejadas del pixel central, serían bastante diferente entre ambos casos.

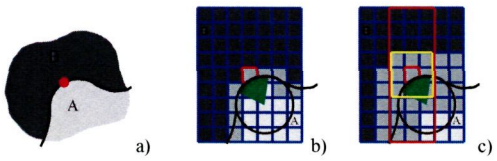


Figura 6.15: a) contorno con curvatura alta; b) imagen digitalizada; c) imagen suavizada. En ambas imágenes se ha dibujado una circunferencia tangente al punto con igual radio que el radio de curvatura en dicho punto.

Sin embargo, usando una ventana $3x3$ (como la marcada en amarillo en la figura), la diferencia sería muy pequeña, y la información que podríamos calcular sobre el contorno sería mucho más precisa. Además, la causa por la que se empezó a usar ventanas tan alargadas era precisamente para garantizar que el contorno la cruzara

siempre de izquierda a derecha. Pero teniendo en cuenta que ahora vamos a considerar también las situaciones en donde esto no se cumple, pues ya no tiene sentido seguir manteniendo estas dimensiones para la ventana.

6.2.1 Detección de circunferencias en una ventana $3x3$

El planteamiento del problema es el siguiente:

Planteamiento Sea F una imagen por cuyo pixel (i,j) pasa un círculo que separa dos zonas de intensidad A (debajo del contorno) y B (encima del contorno). Dicha imagen ha sido suavizada con una máscara de suavizado H_1, tal que

$$H_1 = \begin{pmatrix} a_{11} & a_{01} & a_{11} \\ a_{01} & a_{00} & a_{01} \\ a_{11} & a_{01} & a_{11} \end{pmatrix}$$

dando como resultado la imagen G, donde se cumple que

$$|G_y(i,j)| > |G_x(i,j)|$$

Se desea estimar la expresión de dicho círculo, utilizando sólamente una ventana $3x3$ de la imagen suavizada G centrada en el pixel (i,j).

Solución En principio, podríamos usar el lema 6.1 para estimar la parábola que mejor aproxima las intensidades, utilizando como límites de integración los valores $l_1, m_1, r_1 = -1$ y $l_2, m_2, r_2 = 1$, lo cual daría las expresiones

$$\begin{aligned} c &= \frac{S_L + S_R - 2S_M}{2h(A-B)} \\ b &= \frac{S_R - S_L}{2(A-B)} \\ a &= \frac{2S_M - 3A - 3B}{2(A-B)}h - ch^2\frac{1 + 24a_{01} + 48a_{11}}{12} \end{aligned} \tag{6.8}$$

donde S_L, S_M y S_R son las sumas de las tres columnas de la ventana (ecuación 6.7). A continuación podría usarse el esquena numérico del capítulo anterior (sección 5.3.2) para detectar la circunferencia a partir de la parábola.

El gran problema es que las expresiones para el esquema numérico son incorrectas en el caso en que el contorno no cruce de izquierda a derecha alguna de las columnas. Concretamente, estamos hablando de la que nos devuelve la parábola estimada por el método detector ante la imagen de un círculo (ecuación 5.13). Dicha ecuación partía de que las áreas interiores a cada columna bajo la curva venían dadas por la ecuación (5.12), donde el término U estaba siempre evaluado en los límites izquierdo y derecho de cada columna. Si se quiere tener en cuenta los casos donde la curva pueda cortar la columna por su parte inferior (o superior), habría que deducir el punto de corte, y evaluar la primitiva U en esos puntos, con lo cual saldrían unas expresiones bastante complejas

Sin embargo, disponemos de una solución más práctica para obtener la parábola estimada, sin tener que encontrar su expresión analítica. Usando la misma rutina empleada para generar las ventanas sintéticas a partir de una circunferencia concreta, podemos generar una ventana $5x5$ que represente dicha circunferencia, suavizar su interior para obtener una nueva imagen $3x3$, y aplicar nuestro método detector a dicha subimagen. La parábola obtenida será la parábola estimada que buscábamos. De esta forma podríamos aplicar el mismo esquema numérico que usamos en el capítulo anterior, incluyendo la generación de estas dos subimágenes. El algoritmo sería el siguiente:

	Circunferencia			Parábola estimada			Parábola final		
Iteraciones	R	x_c	y_c	a	b	c	a	b	c
0							−0.1910	0.3142	−0.1448
1	3.9780	1.1924	−3.9861	−0.2006	0.2806	−0.1083	−0.1815	0.3478	−0.1812
2	3.2748	1.0758	−3.2746	−0.1958	0.3100	−0.1334	−0.1768	0.3520	−0.1926
3	3.0932	1.0270	−3.0945	−0.1942	0.3129	−0.1406	−0.1736	0.3533	−0.1967
4	3.0317	1.0099	−3.0322	−0.1924	0.3139	−0.1433	−0.1723	0.3536	−0.1982
5	3.0098	1.0033	−3.0099	−0.1915	0.3142	−0.1443	−0.1718	0.3536	−0.1987
6	3.0026	1.0009	−3.0026	−0.1912	0.3142	−0.1446	−0.1716	0.3536	−0.1988
7	3.0005	1.0002	−3.0006	−0.1911	0.3142	−0.1447	−0.1716	0.3536	−0.1989
8	3.0000	1.0000	−3.0000	−0.1910	0.3142	−0.1448	−0.1716	0.3536	−0.1989

Tabla 6.1: Iteraciones del esquema numérico para encontrar la circunferencia

Funcion EstimarCircunferencia3x3 (P, C)

```
P0 = P
Repetir hasta convergencia
   Deducir la circunferencia C a partir de P (ec.5.10 y 5.11)
   Crear una subimagen F' de 5x5 centrada en el pixel a partir de C
   Suavizar F' para obtener G'
   Calcular la parabola estimada Pe a partir de la imagen G' (ec. 5.13)
   P += P0 - Pe
Fin Repetir
```

6.2.2 Ejemplos sintéticos

En la tabla 6.1 se muestran las iteraciones diferentes para estimar un círculo de radio 3, cuyo centro es el punto $(1, -3)$ (es decir, considerando $(0, 0)$ las coordenadas del pixel cuyo contorno se quiere estimar, el centro del círculo está en el centro geométrico del pixel $(1, 3)$). En la figura 6.16a podemos ver en color negro la circunferencia original que se quiere buscar, en rojo la parábola estimada por nuestro método detector en la primera iteración, y en azul el círculo que comparte posición, pendiente y curvatura con dicha parábola. A medida que avanzan las iteraciones, el círculo azul va convergiendo al que buscamos.

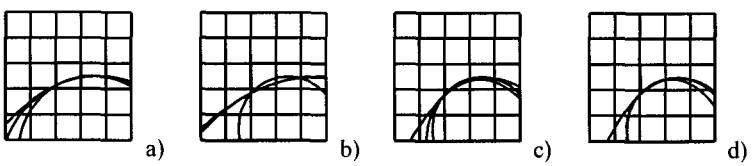


Figura 6.16: Resultado del esquema número. En negro, la circunferencia buscada. En rojo, la parábola estimada. En azul, la circunferencia estimada. a) Círculo de radio 3 centrado en $(1, -3)$, tras la primera iteración. b,c,d) Círculo de radio 2 centrado en $(1, -2)$ tras las iteraciones 1, 10 y 20.

Para círculos de radio inferior el proceso numérico lleva más iteraciones, pero también acaba por converger. En la tabla 6.2 se muestran las iteraciones para un círculo de radio 2 centrado en el punto $(1, -2)$. En la figura 6.16b,c,d se ven los resultados tras 1, 10 y 20 iteraciones.

	Circunferencia			Parábola estimada			Parábola final		
Iteraciones	R	x_c	y_c	a	b	c	a	b	c
0							−0.4328	0.3968	−0.0962
1	6.4700	2.3861	−6.4468	−0.4179	0.3153	−0.0483	−0.4478	0.4782	−0.1442
10	2.4563	1.2014	−2.4852	−0.4419	0.3949	−0.0869	−0.3428	0.5608	−0.3068
20	2.1101	1.0485	−2.1202	−0.4353	0.3962	−0.0936	−0.2890	0.5726	−0.3626
30	2.0301	1.0133	−2.0331	−0.4335	0.3966	−0.0955	−0.2740	0.5760	−0.3785
40	2.0084	1.0037	−2.0093	−0.4330	0.3967	−0.0960	−0.2697	0.5770	−0.3831
50	2.0024	1.0010	−2.0026	−0.4329	0.3967	−0.0962	−0.2684	0.5772	−0.3844
60	2.0007	1.0003	−2.0007	−0.4329	0.3968	−0.0962	−0.2681	0.5773	−0.3848
70	2.0002	1.0001	−2.0002	−0.4328	0.3968	−0.0962	−0.2680	0.5773	−0.3849

Tabla 6.2: Iteraciones del esquema numérico para encontrar la circunferencia

En la figura 6.17 podemos ver visualmente cómo el nuevo esquema permite detectar con bastante precisión radios de curvatura a partir de 2 pixels. Comparando con la detección de la figura 6.13 puede apreciarse la mejoría. Incluso en presencia de ruido, el resultado es bastante satisfactorio. Lógicamente, cuanto más alta sea la curvatura a detectar, más sensibilidad al ruido existirá. Por eso, en la figura el ruido añadido es de magnitud 30, 20 y 10 para los círculos de radio 4, 3 y 2 respectivamente.

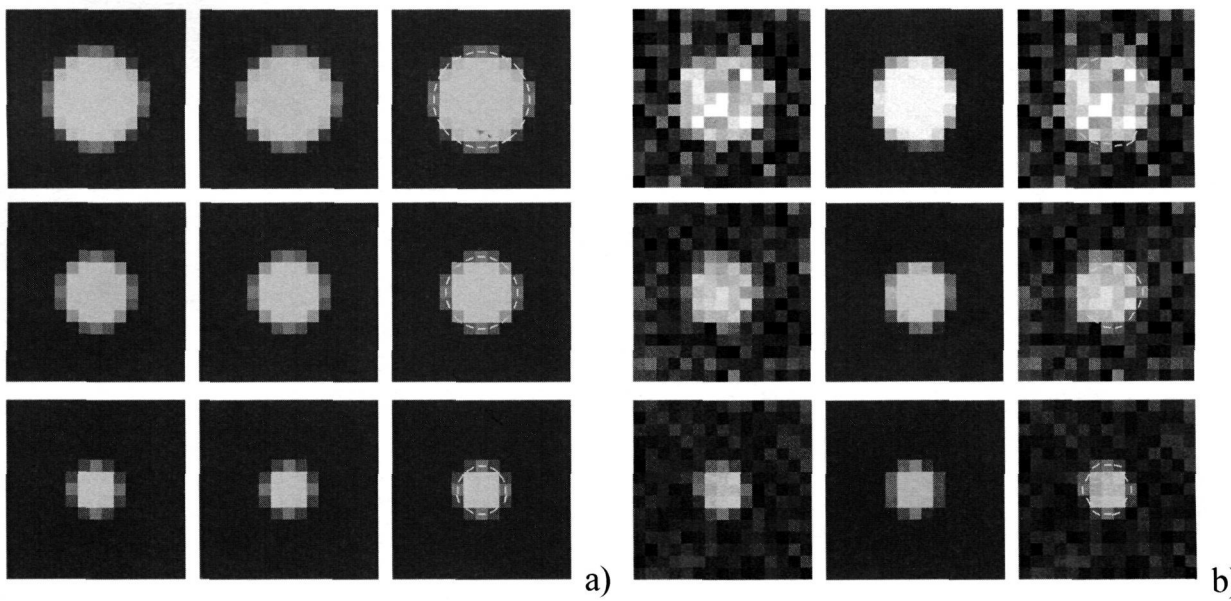


Figura 6.17: De arriba a abajo, el radio del círculo es 4, 3 y 2 pixels. a) de izquierda a derecha: imagen original, imagen restaurada, y contornos estimados. b) igual pero añadiendo ruido

Para radios inferiores a 2 pixels, ya el proceso numérico no siempre converge, y cuando lo hace es después de un número elevado de iteraciones. La razón es que cuanto menor sea el radio de curvatura, mayor es la diferencia entre la circunferencia (que prácticamente cabe entera dentro de la ventana $5x5$) y la parábola estimada. En estos casos, si el contorno es cerrado, representando una pequeña área sobre un fondo de distinta intensidad,

dicha área acaba por desaparecer, ya que prácticamente es como si fuera un ruido. Si por el contrario el contorno es parte de una línea más larga, por ejemplo una esquina, ésta se redondea ligeramente pero respeta la forma básica (ver figura 6.18).

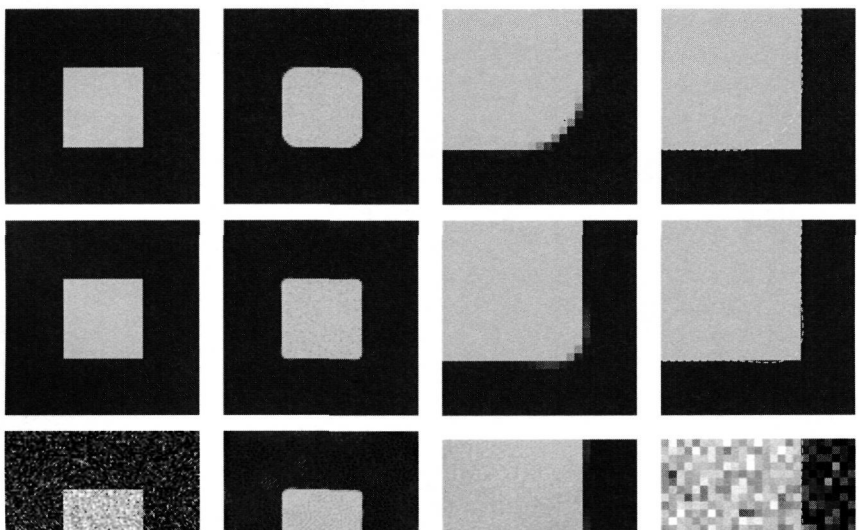


Figura 6.18: De arriba a abajo: restauración básica de un cuadrado ideal, y restauración optimizada para curvaturas altas de un cuadrado ideal y con ruido. De izquiera a derecha: imagen original, imagen restaurada, detalle de la restaurada, y contornos estimados

6.2.3 Algoritmo final

Este esquema de detección de curvaturas altas a partir de subimágenes $3x3$ debe también combinarse con el esquema propuesto para la detección de contornos muy cercanos (sección 6.1.2). La respuesta es sencilla: centrándonos de nuevo en una imagen angiográfica, puede ocurrir que una vena de muy poco grosor tenga además una curvatura alta (como los ejemplos sintéticos de la figura 6.19). Para poder detectar estos casos, debemos realizar conjuntamente ambos procesos.

La solución propuesta es la siguiente: para la estimación de A y B haremos igual que en el algoritmo previo, es decir, buscando en cada columna donde se alcanza un mínimo de la parcial, y usando esos pixels para la estimación. Además dichos límites nos servirán para decidir si existe algún otro contorno muy cercano al que estamos tratando, tanto por arriba como por abajo. En este caso generaremos una subimagen sintética donde aplicar el detector. Finalmente, aplicaremos la detección propuesta en esta sección a partir de una subimagen $3x3$. El algoritmo final propuesto es como sigue:

Funcion ActualizarImagenesFinal (C, I, i, j, orientacion)

```
Segun orientacion vertical / horizontal:
ContornoSuperior = ContornoInferior = False
```

```
Calcular pixels limites l1,m1,r1,l2,m2,r2
Si existe un segundo contorno muy cercano por arriba
   ContornoSuperior = True
   Estimar B a partir de la imagen F
Fin Si
Si existe un segundo contorno muy cercano por debajo
   ContornoInferior = True
   Estimar A a partir de la imagen F
Fin Si
Si ContornoSuperior o ContornoInferior
   Crear una subimagen F' centrada en (i,j) copiando los pixels de F
   Si ContornoSuperior actualizar B en la zona superior de F'
   Si ContornoInferior actualizar A en la zona inferior de F'
   Suavizar F' para obtener G'
   l1 = m1 = r1 = -1
   l2 = m2 = r2 = 1
   Calcular parabola P segun Lema 6.1 a partir de la imagen G'
Si No
   l1 = m1 = r1 = -1
   l2 = m2 = r2 = 1
   Calcular parabola P segun Lema 6.1 a partir de la imagen G
Fin Si
EstimarCircunferencia3x3 (P, C)
Para todos los pixels (m,n) incluidos en los limites de la sumatoria centrada en (i,j)
   Intensidad = Calcular intensidad del pixel (m,n) a partir de la circunferencia C
   Segun el valor de m:
      i:        I(m,n) += 100 * Intensidad
                C(m,n) += 100
      i+1, i-1: I(m,n) += Intensidad
                C(m,n) += 1
   Fin Segun
Fin Para
```

En la figura 6.19 se muestran varios ejemplos sintéticos de venas con grosor 2 y distintos radios de curvatura, con y sin ruido. Puede apreciarse que los contornos son detectados con precisión en las imágenes sin ruido, y en las que sí lo tienen el error cometido es mayor pero no demasiado.

6.2.4 Ejemplos reales

Si volvemos a la imagen del popeye de la figura 6.12 y aplicamos el último esquema propuesto, obtendremos los resultados de la figura 6.20. Vemos que los resultados son más exactos que anteriormente, sobre todo en los puntos con curvatura alta, como las esquinas de la mesa y de la farola, o los codos del muñeco. Los botones de la camisa aparecen más redondeados que en la figura 6.12, y el tornillo del abridor de la ventana mantiene intacto su forma circular, aún teniendo un radio aproximado de dos pixels.

El gran inconveniente de este método es que el tiempo de computo de cada iteración es bastante mayor que antes, ya que para cada pixel borde puede tener que realizar 100 ó 200 iteraciones (dependiendo del número máximo que fijemos para la convergencia), y en cada una de ellas debe generar una subimagen de 25 pixels a partir de la expresión de una circunferenca, suavizarla, y aplicar de nuevo el método detector.

En las figuras 6.21 y 6.22 se ven los resultados de aplicar el método en imágenes angiográficas del riñón y

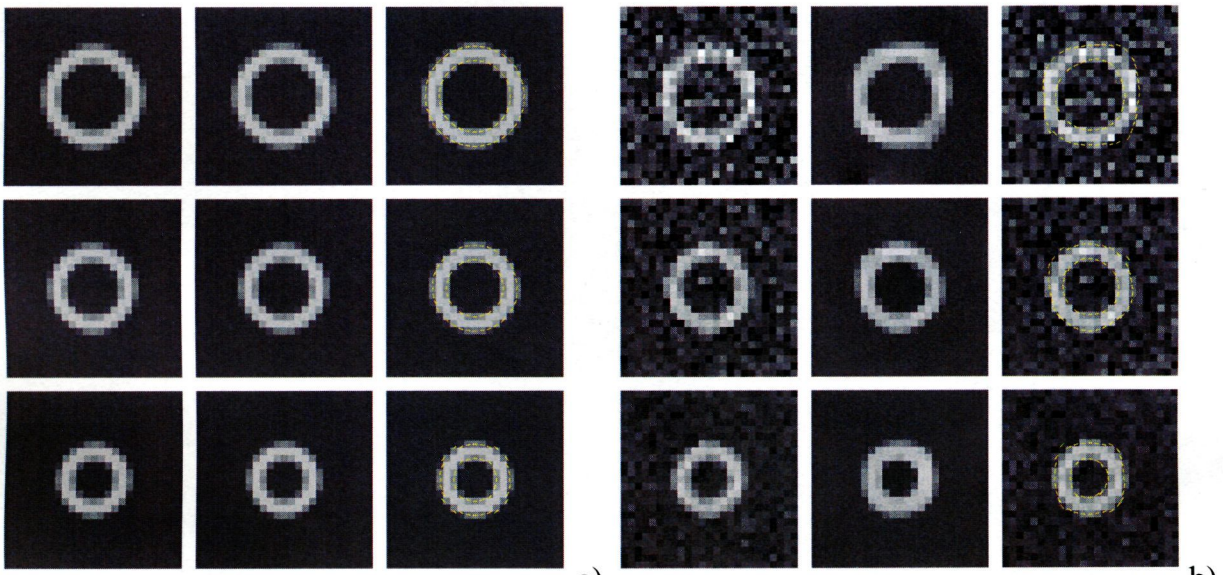


Figura 6.19: De arriba a abajo, los radios de curvatura (exterior / interior) son (7 / 5), (6 / 4), y (5 / 3). a) de izquierda a derecha: imagen original, imagen restaurada, y contornos estimados. b) igual pero añadiendo ruido

del corazón respectivamente. A la vista de los resultados podemos concluir que la localización de los contornos con precisión sub-pixel ha sido bastante precisa en la mayoría de los casos.

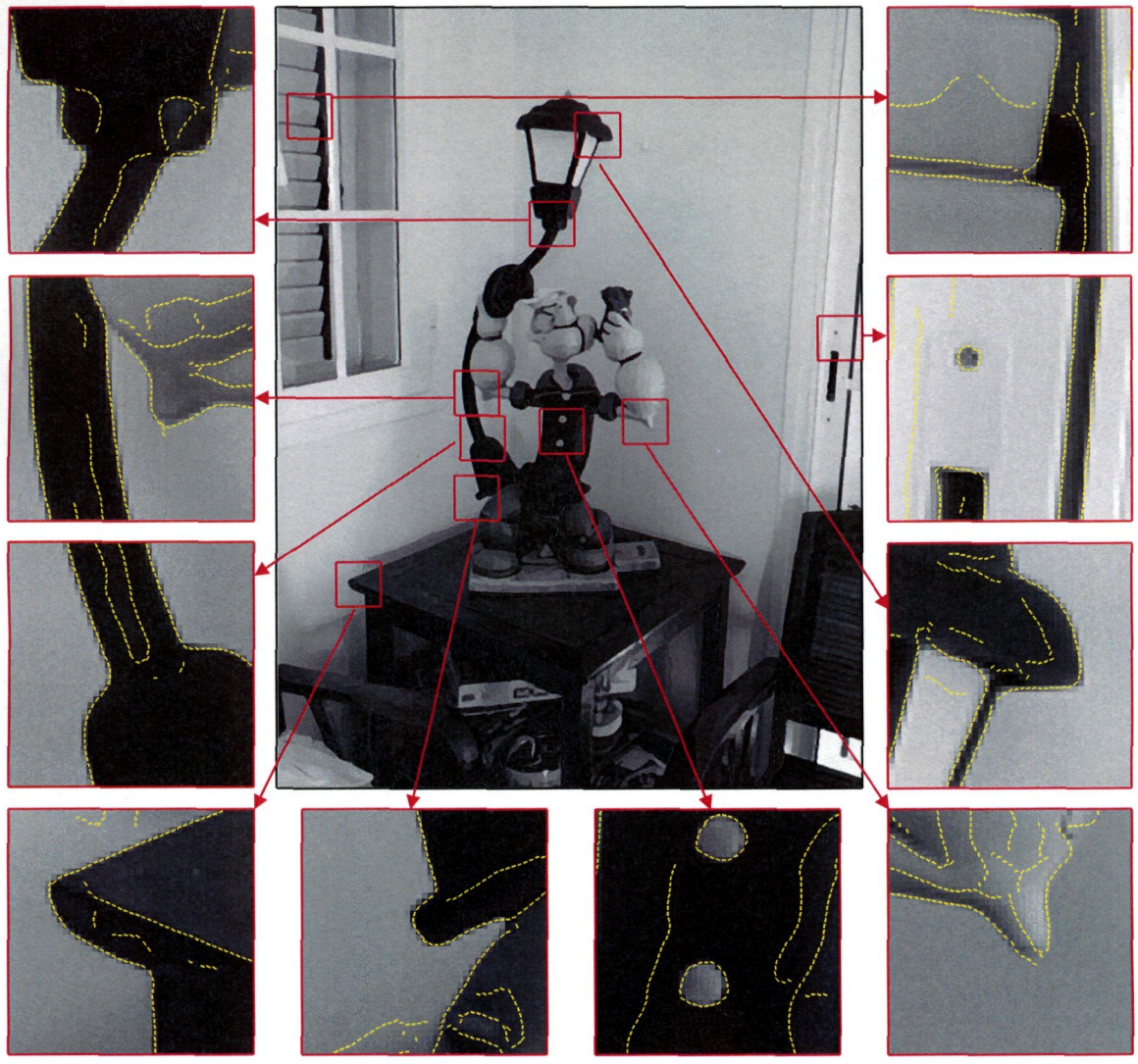


Figura 6.20: Restauración para altas curvaturas en la imagen del popeye

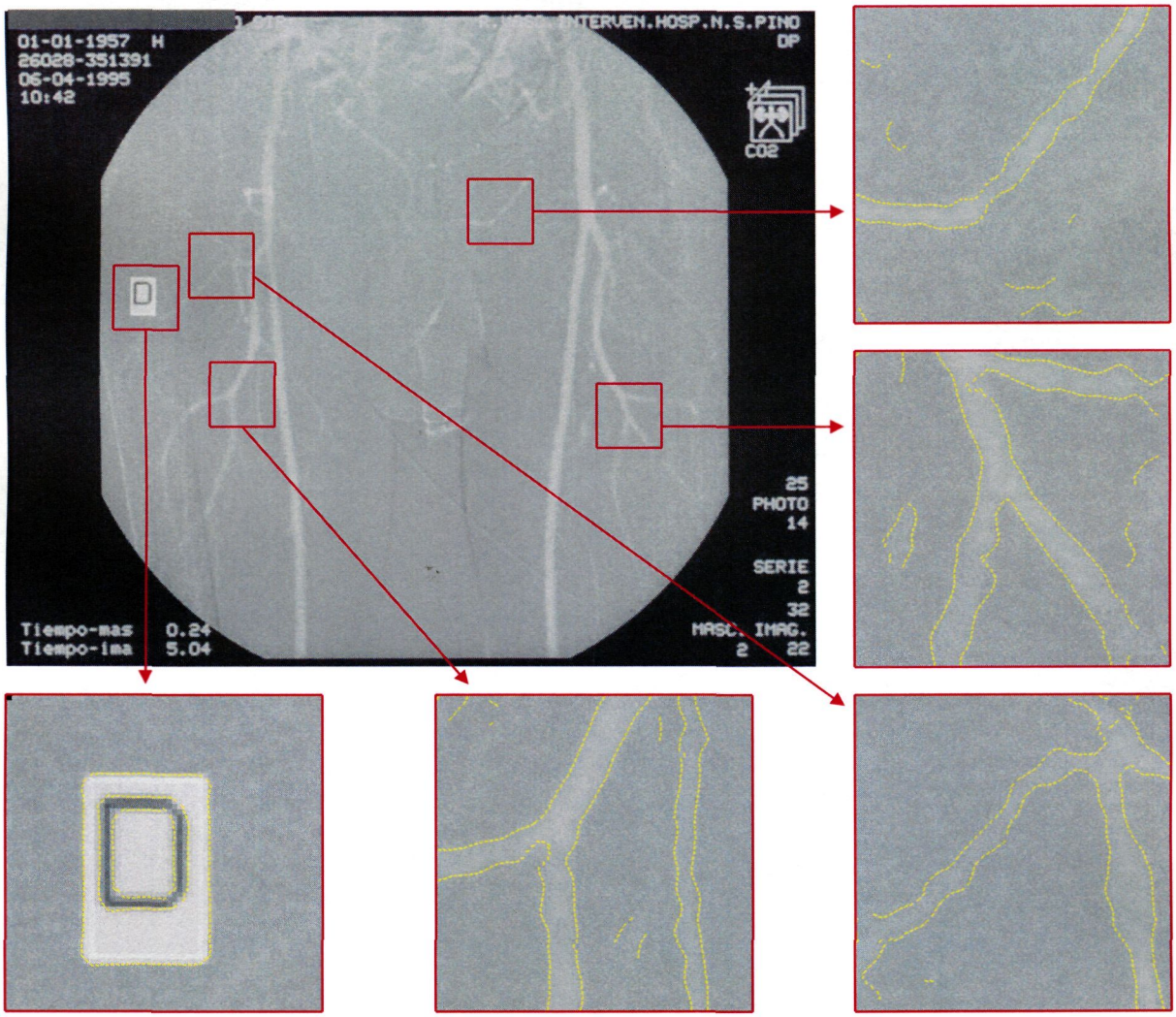


Figura 6.21: Contornos estimados en una angiografía del riñón

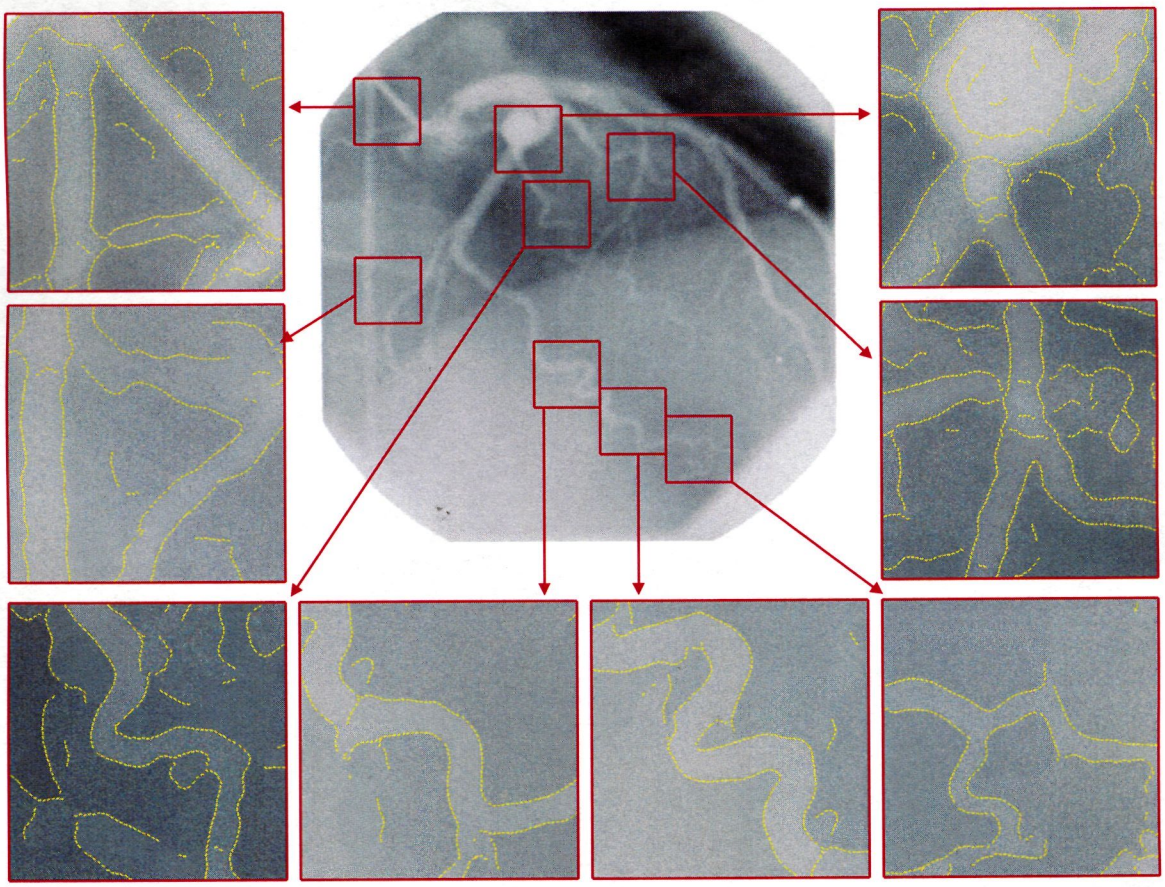


Figura 6.22: Contornos estimados en una angiografía del corazón

Parte II

Localización de contornos en imágenes 3D

Capítulo 7

Contornos de primer orden

- Lindo camino hasta aquí. ¿Crees que podremos extrapolar todo lo anterior para imágenes en 3 dimensiones?
- La primera diferencia es que un contorno no es una línea, sino una superficie.
- Pues empecemos buscando planos.....

ENERO 2003

A lo largo de los capítulos de este bloque, y partiendo de todos los lemas y métodos desarrollados en el bloque previo para imágenes 2D, trataremos de lograr objetivos similares en imágenes tridimensionales. Al entrar a trabajar en un espacio 3D, aparecerán conceptos e ideas nuevas que no existían en el plano 2D, y que iremos viendo a medida que vayan surgiendo.

En los últimos años, la imagen 3D ha tenido un auge espectacular, debido fundamentalmente a dos razones: en primer lugar, la tremenda evolución de la tecnología, que ha permitido incrementar la potencia de cálculo de los procesadores, aumentar la capacidad de almacenamiento de datos en memoria, y desarrollar tarjetas gráficas con prestaciones inimaginables hace una década. Y en segundo lugar, la aparición de máquinas cada vez más sofisticadas, sobre todo dentro del campo de la medicina, que permiten la adquisición de este tipo de imágenes tridimensionales.

El objetivo sigue siendo el mismo: tratar de detectar formas y medir características dentro de la imagen, a partir de la información de los bordes presentes en ella. Por lo tanto, en este primer capítulo del bloque, comenzaremos viendo la forma clásica de detección de bordes en imágenes tridimensionales y sus carencias, para luego desarrollar un algoritmo de detección de bordes de primer orden similar al de 2D, a partir de la información de las imágenes de derivadas parciales. Finalmente, lo aplicaremos a varios ejemplos sintéticos para contrastar sus resultados.

7.1 Técnicas convencionales para la detección de bordes

En primer lugar es preciso definir qué consideramos una imagen 3D. Tal y como mencionan algunos autores ([LOH98, NIK01]), una imagen 3D es una matriz de tres dimensiones cuyos elementos (llamados elementos de volumen o **voxels**) representan muestras de una cierta magnitud física adquirida sobre una malla tridimensional. La manera más común de obtener una imagen 3D es mediante un dispositivo de tomografía, capaz de seccionar un objeto de forma no invasiva. Con esta técnica, los datos obtenidos pueden considerarse como una secuencia

de imágenes 2D paralelas a lo largo de un eje, formando un volumen tridimensional, tal y como se muestra en la figura 7.1. Como el espaciado entre imágenes es bastante pequeño, el valor de cada pixel puede considerarse como la medida de un cierto volumen (con forma de prisma rectangular) dentro de la malla tridimensional.

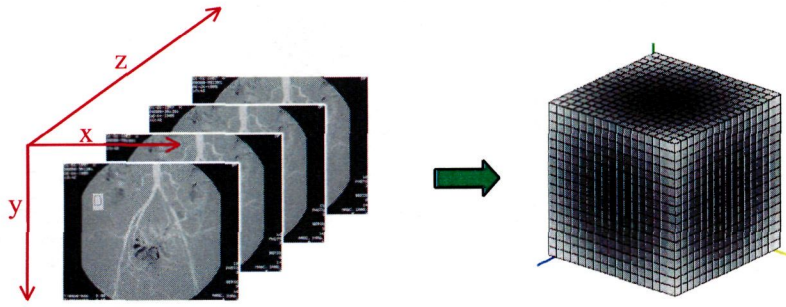


Figura 7.1: Una imagen 3D es una secuencia de imágenes 2D a lo largo de un eje, la cual puede interpretarse como una matriz tridimensional de pixels.

Por lo tanto, al igual que en el caso 2D una imagen puede representarse como una cierta función bidimensional $f(x, y)$ que mide la intensidad en cada punto del plano, podemos representar una imagen 3D como una cierta función tridimensional $f(x, y, z)$ que mide la intensidad en cada punto del espacio. De esta forma, seguiremos considerando borde a aquella zona donde haya una variación brusca de la intensidad. La primera gran diferencia con respecto al caso 2D es que, si anteriormente los pixels borde se unían formando líneas que llamábamos contornos, ahora los voxels borde formarán superficies, las cuales, teóricamente, van a coincidir con la superficies de los objetos que hayan sido adquiridos en la imagen.

7.1.1 El gradiente como herramienta de detección

La técnica más básica para detectar bordes sigue siendo calcular la derivada de la función y buscar máximos locales en la dirección del gradiente [ZHA93]. Para estimar el vector gradiente en un voxel se usarán esquemas parecidos al caso 2D (ver sección 1.1). Es decir, primero se obtiene de forma discreta el valor de las tres derivadas parciales en dicho voxel y, finalmente, se combinan para producir el vector gradiente. Para un caso continuo, la expresión del vector gradiente es la siguiente:

$$\nabla f(x, y, z) = \left[\frac{\partial f(x, y, z)}{\partial x}, \frac{\partial f(x, y, z)}{\partial y}, \frac{\partial f(x, y, z)}{\partial z} \right]$$

Este vector nos indica la dirección de máxima variación en cada punto, y su módulo representa la magnitud de dicha variación. Esto significa que el vector gradiente tendrá magnitud pequeña sobre los puntos de las zonas homogéneas, y tendrá un valor muy alto justo en la zona donde se produce la transición de una intensidad a otra. Además, la dirección de dicho vector nos indicará en cada punto en qué dirección se produce ese cambio.

Tomemos como ejemplo la imagen 3D de una esfera maciza, donde todos los puntos pertenecientes a ella, tanto de su superficie como del interior, aparezcan con una intensidad más clara que los puntos de exterior. Dicha imagen 3D tendría gradiente muy bajo en toda la zona exterior e interior a la esfera, y un vector gradiente con magnitud alta a lo largo de toda la superficie, cuya dirección estaría apuntando en todos los puntos hacia el centro de la esfera.

Por tanto, la técnica más básica para detectar bordes consiste en evaluar el vector gradiente en cada voxel, y quedarnos con aquéllos cuyas magnitudes sean máximos locales en la dirección de dicho vector. Pero para poder

evaluar el gradiente, necesitamos conocer cómo discretizar las derivadas, ya que nuestra imagen de entrada no será una señal continua sino discreta, debido al proceso de adquisición.

7.1.2 Discretización de la derivada parcial de una función $f(x, y, z)$.

Sea $f(x, y, z)$ una función de $\mathbb{R}^3$ en $\mathbb{R}$. Para estimar el valor de la derivada parcial en la dirección de la variable x en un punto (x_0, y_0, z_0) a partir del valor de la función en puntos cercanos, podemos usar el desarrollo de Taylor de la función en los puntos $(x_0 + h_x, y_0, z_0)$ y $(x_0 - h_x, y_0, z_0)$, de la siguiente manera:

$$\begin{aligned} f(x_0 + h_x, y_0, z_0) &= f(x_0, y_0, z_0) + h_x f_x(x_0, y_0, z_0) + O(h_x^2) \\ f(x_0 - h_x, y_0, z_0) &= f(x_0, y_0, z_0) - h_x f_x(x_0, y_0, z_0) + O(h_x^2) \end{aligned}$$

donde h_x es un valor relativamente pequeño. Restando ambas expresiones obtenemos que

$$f(x_0 + h_x, y_0, z_0) - f(x_0 - h_x, y_0, z_0) = 2h_x f_x(x_0, y_0, z_0) + O(h_x^2)$$

de donde se deduce que, para valores pequeños de h_x, la derivada parcial en x en el punto (x_0, y_0, z_0) puede aproximarse por

$$f_x(x_0, y_0, z_0) \simeq \frac{f(x_0 + h_x, y_0, z_0) - f(x_0 - h_x, y_0, z_0)}{2h_x}$$

Al igual que ocurría con las funciones de dos variables, esto significa que, si la función $f(x, y, z)$ ha sido discretizada en una muestra tridimensional $\{..., f(x_i, y_j, z_k), ...\}$, siendo $i, j, k \in Z$, donde la distancia horizontal h_x entre puntos vecinos puede considerarse pequeña, y de tal forma que desconocemos los valores de la función en ningún otro lugar, podemos estimar la derivada parcial en la dirección x en el punto (x_i, y_j, z_k) a partir de los valores de la función en sus dos puntos vecinos, (x_{i-1}, y_j, z_k) y (x_{i+1}, y_j, z_k), mediante la siguiente expresión:

$$f_x(x_i, y_j, z_k) \simeq \frac{f(x_{i+1}, y_j, z_k) - f(x_{i-1}, y_j, z_k)}{2h_x} \tag{7.1}$$

Lógicamente, al haber discretizado la función, esto no es más que una estimación del verdadero valor de la derivada. Otra estimación alternativa que puede dar mejores resultados es usar también la anterior expresión pero aplicada a los 4 puntos vecinos (x_i, y_{j-1}, z_k), (x_i, y_{j+1}, z_k), (x_i, y_j, z_{k-1}) y (x_i, y_j, z_{k+1}). De esta manera, el valor final que daremos para la derivada parcial en la dirección x en el punto (x_i, y_j, z_k) será un promedio entre las expresiones individuales de las cinco filas. Por simplificar la expresión, sea $F_{i,j,k} = f(x_i, y_j, z_k)$, y consideremos como punto de interés $(x_0, y_0, z_0) = (0, 0, 0)$, con lo cual obtenemos que

$$\begin{aligned} f_x(0,0,0) &= (1-\alpha)\left(\frac{F_{1,0,0} - F_{-1,0,0}}{2h_x}\right) + \\ &+ \frac{\alpha}{4}\left(\frac{F_{1,-1,0} - F_{-1,-1,0} + F_{1,1,0} - F_{-1,1,0} + F_{1,0,-1} - F_{-1,0,-1} + F_{1,0,1} - F_{-1,0,1}}{2h_x}\right) \end{aligned}$$

donde $0 \leqslant \alpha \leqslant 1$. Con el valor α controlamos el peso que queremos darle a las filas adyacentes. Si $\alpha = 0$ sólo consideramos la fila actual, y estaríamos en el caso inicial. Teniendo en cuenta que la distancia entre puntos es pequeña, lo que en realidad estamos haciendo con esta nueva expresión es promediar la derivada en un entorno alrededor del punto, disminuyendo así el posible ruido que podríamos haber obtenido en la discretización.

De hecho, podrían considerarse aún más filas tomando las vecinas en diagonal, es decir, usar también la expresión 7.1 para los puntos $(0, 1, 1)$, $(0, 1, -1)$, $(0, -1, 1)$ y $(0, -1, -1)$, y asignarles un segundo peso diferente β, donde se cumple que $0 \leqslant \beta \leqslant 1$ y $\beta \leq \alpha$. Así, el promedio del valor sería aún mayor. La expresión final quedaría como sigue:

$$
\begin{aligned}
f_x(0,0,0) &= (1-\alpha-\beta)\left(\frac{F_{1,0,0}-F_{-1,0,0}}{2h_x}\right)+ \\
&+\frac{\alpha}{4}\left(\frac{(F_{1,-1,0}-F_{-1,-1,0})+(F_{1,1,0}-F_{-1,1,0})+(F_{1,0,-1}-F_{-1,0,-1})+(F_{1,0,1}-F_{-1,0,1})}{2h_x}\right)+ \\
&+\frac{\beta}{4}\left(\frac{(F_{1,1,1}-F_{-1,1,1})+(F_{1,1,-1}-F_{-1,1,-1})+(F_{1,-1,1}-F_{-1,-1,1})+(F_{1,-1,-1}-F_{-1,-1,-1})}{2h_x}\right)
\end{aligned}
$$

Para calcular la derivada parcial en las otras direcciones y y z se procede exactamente igual, intercambiando el nombre de las variables. Las expresiones serían:

$$
\begin{aligned}
f_y(0,0,0) &= (1-\alpha-\beta)\left(\frac{F_{0,1,0}-F_{0,-1,0}}{2h_y}\right)+ \\
&+\frac{\alpha}{4}\left(\frac{(F_{-1,1,0}-F_{-1,-1,0})+(F_{1,1,0}-F_{1,-1,0})+(F_{0,1,-1}-F_{0,-1,-1})+(F_{0,1,1}-F_{0,-1,1})}{2h_y}\right)+ \\
&+\frac{\beta}{4}\left(\frac{(F_{1,1,1}-F_{1,-1,1})+(F_{1,1,-1}-F_{1,-1,-1})+(F_{-1,1,1}-F_{-1,-1,1})+(F_{-1,1,-1}-F_{-1,-1,-1})}{2h_y}\right) \\
f_z(0,0,0) &= (1-\alpha-\beta)\left(\frac{F_{0,0,1}-F_{0,0,-1}}{2h_z}\right)+ \\
&+\frac{\alpha}{4}\left(\frac{(F_{-1,0,1}-F_{-1,0,-1})+(F_{1,0,1}-F_{1,0,-1})+(F_{0,-1,1}-F_{0,-1,-1})+(F_{0,1,1}-F_{0,1,-1})}{2h_z}\right)+ \\
&+\frac{\beta}{4}\left(\frac{(F_{1,1,1}-F_{1,1,-1})+(F_{1,-1,1}-F_{1,-1,-1})+(F_{-1,1,1}-F_{-1,1,-1})+(F_{-1,-1,1}-F_{-1,-1,-1})}{2h_z}\right)
\end{aligned}
$$

Como puede apreciarse, se ha tenido en cuenta que la distancia entre muestras en cada dimensión (distancia entre voxels consecutivos) pueda no ser la misma. En la práctica, $h_x = h_y$ en todos los casos, siendo h_z la que puede diferir. En el presente trabajo consideraremos siempre la situación más general, donde cada h es independiente de las otras.

7.1.3 Máscaras para el cálculo de las derivadas parciales

Con la discretización obtenida, el proceso para calcular las derivadas parciales es equivalente a convolucionar la imagen usando tres máscaras tridimensionales. Para poder notar una matriz tridimensional, sigamos el criterio indicado en la figura 7.2, donde una matriz 3D M centrada en el voxel $(0,0,0)$ se representa en la forma

$$M = [..., M_{-1}, M_0, M_1, ...]$$

siendo M_k una matriz 2D cuyos valores representan los voxels de la sección $z = k$.

Con esta notación, las tres máscaras que nos devuelven el valor de las derivadas parciales son las siguientes

$$
H_x = \frac{1}{2h_x}\left[\begin{bmatrix} -\beta/4 & 0 & \beta/4 \\ -\alpha/4 & 0 & \alpha/4 \\ -\beta/4 & 0 & \beta/4 \end{bmatrix}, \begin{bmatrix} -\alpha/4 & 0 & \alpha/4 \\ -(1-\alpha-\beta) & 0 & (1-\alpha-\beta) \\ -\alpha/4 & 0 & \alpha/4 \end{bmatrix}, \begin{bmatrix} -\beta/4 & 0 & \beta/4 \\ -\alpha/4 & 0 & \alpha/4 \\ -\beta/4 & 0 & \beta/4 \end{bmatrix}\right] \tag{7.2}
$$

$$
H_y = \frac{1}{2h_y}\left[\begin{bmatrix} -\beta/4 & -\alpha/4 & -\beta/4 \\ 0 & 0 & 0 \\ \beta/4 & \alpha/4 & \beta/4 \end{bmatrix}, \begin{bmatrix} -\alpha/4 & -(1-\alpha-\beta) & -\alpha/4 \\ 0 & 0 & 0 \\ \alpha/4 & (1-\alpha-\beta) & \alpha/4 \end{bmatrix}, \begin{bmatrix} -\beta/4 & -\alpha/4 & -\beta/4 \\ 0 & 0 & 0 \\ \beta/4 & \alpha/4 & \beta/4 \end{bmatrix}\right]
$$

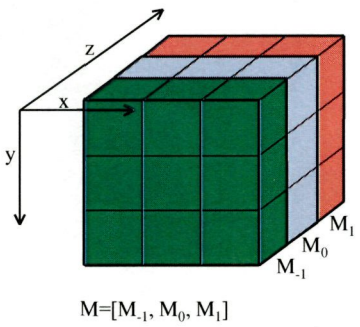


Figura 7.2: Notación de una matriz $3 \times 3 \times 3$

$$H_z = \frac{1}{2h_z}\left[\begin{bmatrix} -\beta/4 & -\alpha/4 & -\beta/4 \\ -\alpha/4 & -(1-\alpha-\beta) & -\alpha/4 \\ -\beta/4 & -\alpha/4 & -\beta/4 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} \beta/4 & \alpha/4 & \beta/4 \\ \alpha/4 & (1-\alpha-\beta) & \alpha/4 \\ \beta/4 & \alpha/4 & \beta/4 \end{bmatrix}\right]$$

donde se ha supuesto que el sentido de los ejes direccionales es el que aparece en la figura 7.2 (la x crece hacia la derecha, la y hacia abajo y la z hacia dentro).

Revisando la bibliografía [LOH98], vemos que muchas de las máscaras usadas tradicionalmente caen dentro de este esquema, siendo meras extensiones de las máscaras usadas en 2D. Así tenemos la máscara de Prewit, donde todas las filas tienen el mismo peso, siendo $\alpha = \beta = 4/9$, cuya expresión es

$$H_z = \frac{1}{18h_z}\left[\begin{bmatrix} -1 & -1 & -1 \\ -1 & -1 & -1 \\ -1 & -1 & -1 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}\right]$$

Sobel usa $\alpha = 8/15$ y $\beta = 4/15$ lo cual produce la máscara

$$H_z = \frac{1}{30h_z}\left[\begin{bmatrix} -1 & -2 & -1 \\ -2 & -3 & -2 \\ -1 & -2 & -1 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 2 & 1 \\ 2 & 3 & 2 \\ 1 & 2 & 1 \end{bmatrix}\right]$$

donde las filas vecinas en las direcciones principales tienen el doble de peso que las vecinas diagonales, y la fila central el triple.

Por otro lado, Zucker [ZUC81] propone la máscara

$$H_z = \frac{1}{2(4a+4b+c)h_z}\left[\begin{bmatrix} -b & -a & -b \\ -a & -c & -a \\ -b & -a & -b \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} b & a & b \\ a & c & a \\ b & a & b \end{bmatrix}\right]$$

donde $a = 1$, $b = \sqrt{2}/2$ y $c = \sqrt{3}/3$, más en la línea de la máscara 2D propuesta por Frey y Chen (ecuación 1.5) para mantener la condición de que los bordes diagonales perfectos tengan la misma magnitud del gradiente que los bordes en las direcciones principales, aunque hay que mencionar que su interpretación de "bordes perfectos" sigue siendo diferente a la que proponemos en las hipótesis iniciales de esta tesis.

Por último, L. Álvarez y otros [ALV01] hacen un estudio más profundo en este aspecto, dando su propia definición de "borde perfecto". Para ellos, un borde viene dado por una función

$$F(x,y,z) = \begin{cases} 1 & \text{si} \quad ax+by+cz \leq 0 \\ 0 & \text{si} \quad ax+by+cz > 0 \end{cases}$$

donde (x, y, z) son las coordenadas enteras de un voxel, y el plano $ax + by + cz = 0$ indica la orientación del borde. Para obtener el valor de los coeficientes α y β imponen que la norma del gradiente calculado con las máscaras sea la misma en el punto central $(0, 0, 0)$ para cualquier elección de a, b y c, donde $a, b, c \in \{-1, 0, 1\}$. Los valores obtenidos son

$$\begin{aligned} \alpha &= 2\sqrt{2} - \frac{4}{3}\sqrt{3} \simeq 0.519 \\ \beta &= 2 - 2\sqrt{2} + \frac{2}{3}\sqrt{3} \simeq 0.326 \end{aligned} \tag{7.3}$$

Existen otras máscaras de distinto tamaño [ZHA93], como las de 2x2x2, que tienen el problema de no estar centradas en el voxel que se está tratando, o de 5x5x5, que exigen un costo computacional mucho mayor, aunque al tomar un entorno más amplio, son menos sensibles al ruido, y obtienen un resultado más suavizado (y por lo tanto a veces menos preciso). Pero la idea subyacente en todas ellas es la misma: obtener estimaciones para los valores de la derivada para, a partir de ahí, encontrar los bordes de la imagen.

7.1.4 Cálculo del vector gradiente

Una vez estimadas las derivadas parciales en un punto de la imagen, podemos estimar el vector gradiente en dicho punto. Sea $F(i, j, k)$ la imagen original, y sean $F_x(i, j, k)$, $F_y(i, j, k)$ y $F_z(i, j, k)$ las tres imágenes resultantes de aplicar la convolución con las máscaras H_x, H_y y H_z respectivamente. El vector gradiente para cada voxel (i, j, k) será entonces

$$\nabla F(i, j, k) = [F_x(i, j, k), F_y(i, j, k), F_z(i, j, k)]$$

A partir de esta estimación, podemos obtener una imagen de gradientes (tomando únicamente el módulo) de la manera siguiente:

$$G(i, j, k) = \sqrt{F_x^2(i, j, k) + F_y^2(i, j, k) + F_z^2(i, j, k)}$$

Con dicha imagen podremos estimar que en aquellos puntos (i, j, k) donde el valor de $G(i, j, k)$ sea suficientemente grande, existirá un borde cuya orientación será perpendicular a la dirección del vector $[F_x(i, j, k), F_y(i, j, k), F_z(i, j, k)]$.

En la figura 7.3 podemos ver un ejemplo de imagen de gradientes, aplicado sobre la imagen de una esfera de radio 20, donde los puntos de su superficie y del interior tienen una intensidad más clara que los puntos del exterior. Las tres primeras imágenes de la izquierda corresponden con los valores de las imágenes de parciales en x, y y z respectivamente, usando las máscaras anteriores para cada derivada, y la última imagen de la derecha es la del módulo del gradiente. Cada voxel se ha representado como un pequeño cubo, cuya intensidad se corresponde con el valor del voxel correspondiente (en valor absoluto, para tener intensidades positivas). Sólo se han representado los voxels cuyo valor supera un cierto umbral. Es por eso que los voxels pertenecientes a zonas homogéneas (interior y exterior de la esfera) no aparecen (pues su derivada parcial es nula), así como una franja central alrededor de la esfera en las tres imágenes de las parciales, correspondiente a los voxels cuya derivada parcial no es lo suficientemente grande. Puede apreciarse cómo efectivamente los voxels cuyo módulo del gradiente es superior a un cierto umbral representan los voxels pertenecientes a la superficie del objeto que aparece en la imagen.

7.2 Hipótesis de trabajo

Al igual que en el caso de las imágenes 2D, hemos visto que el uso de una herramienta matemática como es el vector gradiente nos aporta bastante información sobre la existencia de bordes en una imagen. Sin embargo, de nuevo hay que mencionar que para poder aplicar esta herramienta con rigor es preciso que nuestra función sea

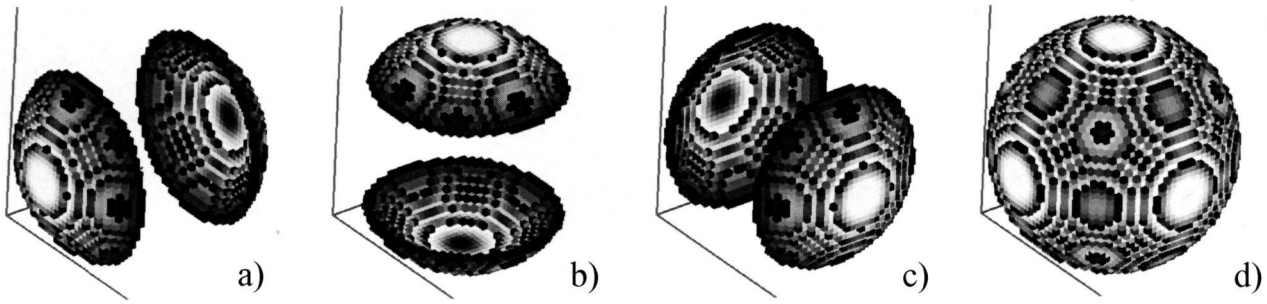


Figura 7.3: Derivadas de una imagen de una esfera de radio 20: a) parcial en x; b) parcial en y; c) parcial en z; d) módulo del gradiente. Los ejes representan las direcciones x (rojo), y (verde) y z (azul)

derivable (y por tanto continua) en todo su dominio (o al menos en aquellos puntos donde queramos evaluar el vector gradiente). Es por ello que todos estos trabajos toman como premisa fundamental el hecho de que, aunque nuestra imagen de entrada sea una señal con valores discretos, dicha señal proviene del muestreo de una cierta función continua y derivable, a la cual pretendemos calcular los bordes.

A partir de aquí vamos a hacer una interpretación diferente. Asumiremos que justo en las zonas donde hay un borde en la imagen, es decir, en las superficies de los objetos presentes en ella, habrá una discontinuidad. La razón es muy simple: tratándose de una imagen volumétrica (perteneciente a $\mathbb{R}^3$) si un objeto macizo tiene intensidad A y está ubicado dentro de un volumen de intensidad B (ver figura 7.4), la función intensidad medida en cada punto del espacio presenta una discontinuidad en todos los puntos de la superficie del objeto.

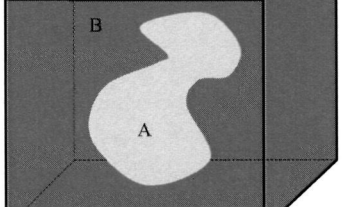


Figura 7.4: Existe una discontinuidad en los puntos de la superficie del objeto de intensidad A inmerso en un volumen de intensidad B

Con este **modelo de borde**, ya no tendrá sentido hablar del "gradiente en un voxel", sino del "vector normal a la superficie que pasa por ese voxel". Y cuando, de ahora en adelante, hablemos de derivadas parciales en un voxel, estaremos refiriéndonos en realidad al resultado de aplicar lás máscaras de convolución anteriores, pero no al valor de la derivada en sí, puesto que damos por hecho que no existe, al no haber continuidad en la función.

La pregunta que surge ahora es relativa al muestreo de la imagen, es decir, al proceso de adquisición: si realmente existe una discontinuidad, ¿qué ocurre con los voxels donde cae justo esa discontinuidad? ¿Qué intensidad tendrán? ¿A? ¿B? Para responder a estas preguntas, usaremos el siguiente **modelo de adquisición**, similar al que hemos usado para el caso 2D: la intensidad final del voxel será un promedio entre ambas intensidades, A y B, ponderados por el volumen relativo que cada intensidad ocupa dentro del voxel, considerando éste como

un pequeño prisma rectangular dentro de la imagen. Es decir, atendiendo a la figura 7.5, el valor de intensidad del voxel seleccionado vendrá dado por la expresión

$$F(i,j,k) = \frac{AV_A + BV_B}{h_x h_y h_z} \tag{7.4}$$

donde A y B son las intensidades de ambas zonas, V_A y V_B son los volúmenes de las regiones ocupadas por ambas intensidades respectivamente en el interior del voxel, y h_x, h_y y h_z son las longitudes de cada lado del voxel, en las direcciones x, y y z respectivamente. Se cumple por tanto que el volumen del voxel es igual a $V_A + V_B = h_x h_y h_z$.

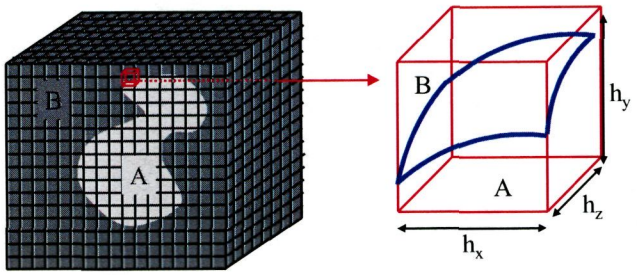


Figura 7.5: La intensidad del voxel seleccionado es un promedio entre A y B ponderado cada uno por el volumen relativo que ocupa en el interior del voxel

La hipótesis planteada no es nueva. Ya existen otros autores que la mencionan [SON00, SUE02], y comúnmente se le denomina **PVE (partial volume effect)**. Este efecto es muy conocido dentro del área de imagen médica. Hay que tener en cuenta que tanto en imágenes de resonancia como en tomografía, cada tejido orgánico produce un cierto valor de intensidad en la imagen. Esto significa que cuando en el interior del volumen correspondiente a un voxel coexisten más de un tipo de tejido diferente, el valor obtenido para dicho voxel será un promedio de los valores individuales producidos por cada tejido.

Existen diferentes modelos para representar este efecto en la literatura [SAN95, RIF99, AST02], dependiendo incluso del tipo de dispositivo de adquisición que se esté usando (CT, MRI), pero nosotros vamos a partir del planteado en la ecuación 7.4, que es lo suficientemente simple como para que pueda funcionar bastante bien en cualquier tipo de imagen 3D. A partir de este modelo desarrollaremos todos los métodos expuestos en éste y en los próximos capítulos.

Errores cometidos por las máscaras H_x, H_y y H_z Con los modelos de borde y adquisición que acabamos de plantear, es fácil ver cómo el método de estimación del vector gradiente a partir de la convolución con las máscaras H_x, H_y y H_z no es exacto en todos los casos. De hecho, en la figura 7.3d, donde se muestra el valor del módulo del vector gradiente para cada voxel de la superficie de la esfera (generada sintéticamente con nuestro modelo de adquisición), éste debería ser igual en todos los voxels[1], y sin embargo puede apreciarse la diferencia de intensidad en cada uno.

Siendo más precisos, podríamos rehacer todos los ejemplos de la sección 1.3 para imágenes 2D, considerando que todas las secciones para un valor de z constante son iguales entre sí, y llegaríamos a resultados similares, es decir, que es imposible estimar con exactitud el vector normal y el salto de intensidad producido en cada voxel

[1] El valor correcto debería ser la diferencia de intensidad entre el interior y el exterior de la esfera.

de una superficie en todos los casos, a partir de las máscaras de convolución para las derivadas parciales, con independencia de los valores que demos a los coeficientes α y β.

En lo que resta de capítulo, vamos a proponer un método que sí sea capaz de estimar con precisión no sólo el vector normal y el salto de intensidad en cada voxel de la superficie, sino también la ubicación de dicha superficie en el interior de cada voxel. Por supuesto, la exactitud total se conseguirá sólo en imágenes ideales (ideal en el sentido de las hipótesis planteadas).

7.3 Cálculo del salto de intensidad

El objetivo es, a partir de los valores de intensidad de la imagen original, y conociendo que el valor de cada voxel ha sido obtenido según la hipótesis planteada en nuestro modelo de adquisición, tratar de obtener una expresión para la superficie real que representa el borde del objeto dentro del voxel. Lógicamente, no podemos encontrar la expresión exacta, puesto que debido al proceso de adquisición, se ha perdido información del comportamiento del borde en el interior del voxel. Por lo tanto, inicialmente, haremos la siguiente simplificación: supondremos que, localmente, la superficie sobre un voxel y su vecindad se comporta como un plano. Para ello, vamos a introducir las dos definiciones siguientes:

Llamaremos **contorno de primer orden** a la función intensidad siguiente

$$f(x,y,z,A,B,a,b,c,e) = \begin{cases} A & \text{si } ax+by+cz+e \geq 0 \\ B & \text{si } ax+by+cz+e < 0 \end{cases}$$

tal que a cada punto $(x,y,z) \in \mathbb{R}^3$ se le asocia una intensidad A o B en función de a qué lado del plano $ax+by+cz+e=0$ se encuentra, tal y como se ve en la figura 7.6a. Y llamaremos **imagen ideal con un contorno de primer orden** a la imagen siguiente

$$F_{i,j,k}(A,B,a,b,c,e) = \int_{z_k-h_z/2}^{z_k+h_z/2} \int_{y_j-h_y/2}^{y_j+h_y/2} \int_{x_i-h_x/2}^{x_i+h_x/2} f(x,y,z,A,B,a,b,c,e)dxdydz \tag{7.5}$$

donde $f(x,y,z,A,B,a,b,c,e)$ es un contorno de primer orden. Esta imagen puede verse en la figura 7.6b. Esto significa que el método que a continuación se desarrollará tratará de, partiendo sólamente de los valores de los voxels de la imagen $F_{i,j,k}$, encontrar los valores exactos para a, b, c, e y $A-B$.

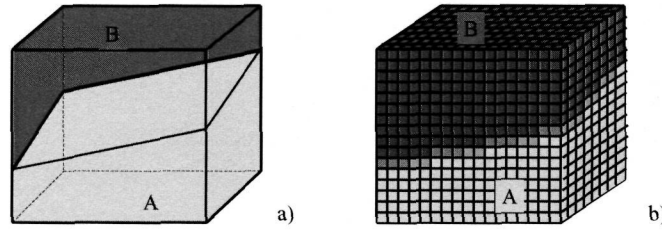


Figura 7.6: a) Contorno de primer orden, $f(x,y,z,A,B,a,b,c,e)$; b) Imagen ideal con un contorno de primer orden, $F_{i,j,k}(A,B,a,b,c,e)$

7.3.1 Caso general para el primer octante

Sea $F_{i,j,k}(A,B,a,b,c,e)$ una imagen ideal con un contorno de primer orden, donde el contorno atraviesa el voxel central de la imagen. Por simplicidad, dicho voxel tendrá coordenadas $(0,0,0)$. Supondremos también que $A > B$.

Inicialmente tomaremos sólo el caso en que el vector normal al plano tenga sus tres componentes mayores o iguales que cero, es decir $a, b, c \geq 0$ en el voxel cuyo contorno queremos estimar. De esta forma nos quedamos sólo con los vectores del primer octante. Los demás octantes se resolverán por simetría. Además, dentro de este caso, sólo vamos a estudiar el caso donde $b > a, c$ en el voxel. Los otros dos casos también se resolverán por simetría. En definitiva, vamos a estudiar sólo el caso donde el vector normal caiga dentro de la zona que se muestra en la figura 7.7a.

En estas condiciones, el plano tendrá una orientación como la de la figura 7.7b, y vendrá definida por dos ángulos, θ_1 y θ_2, donde $\theta_1, \theta_2 \in [0, \pi/4]$. Si calculamos las tres imágenes de parciales, F_x, F_y y F_z, convolucionando con las máscaras H_x, H_y y H_z respectivamente, lo anterior es equivalente a decir que el vector gradiente $[F_x, F_y, F_z]$ en el voxel en cuestión cumpla que $F_y > F_x, F_z \geq 0$.

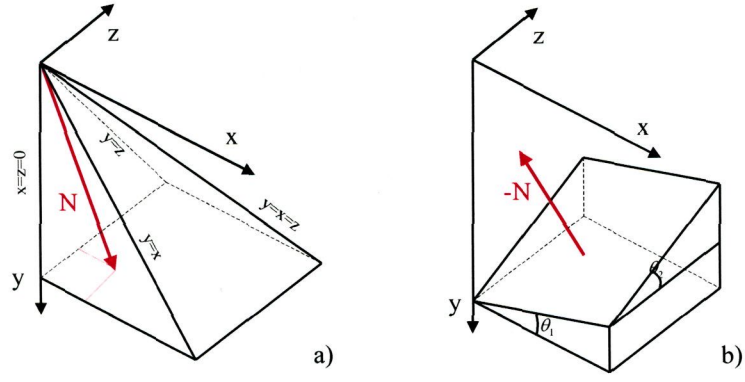


Figura 7.7: Caso en el que $F_y > F_x, F_z \geq 0$: a) Rango de posibilidades para la dirección del vector normal; b) La orientación del plano viene dada por los ángulos $\theta_1, \theta_2 \in [0, \pi/4]$. La normal se muestra invertida

Si trazamos toda la familia de planos posibles para este rango de variación de los ángulos θ_1 y θ_2 que pasen por el voxel central $(0, 0, 0)$ obtenemos la región mostrada en color gris claro en la figura 7.8. Esto significa que todos los voxels que no intersecten con dicha región tendrán valor A (si se encuentran por debajo del borde) o B (si se encuentran por encima), y que aquéllos que sí intersecten tendrán un valor V tal que $B \leq V \leq A$.

7.3.2 Derivadas parciales en la dirección y

Si tomamos la sección $z = 0$ de la imagen F (es decir, la sección central en la figura 7.8 donde aparece el voxel central seleccionado), ésta tendrá la expresión siguiente:

$$F(i,j,0) = \begin{bmatrix} B & B & B & B & F_{1,-3,0} & F_{2,-3,0} & F_{3,-3,0} \\ B & B & B & F_{0,-2,0} & F_{1,-2,0} & F_{2,-2,0} & F_{3,-2,0} \\ F_{-3,-1,0} & F_{-2,-1,0} & F_{-1,-1,0} & F_{0,-1,0} & F_{1,-1,0} & F_{2,-1,0} & F_{3,-1,0} \\ F_{-3,0,0} & F_{-2,0,0} & F_{-1,0,0} & F_{0,0,0} & F_{1,0,0} & F_{2,0,0} & F_{3,0,0} \\ F_{-3,1,0} & F_{-2,1,0} & F_{-1,1,0} & F_{0,1,0} & F_{1,1,0} & F_{2,1,0} & F_{3,1,0} \\ F_{-3,2,0} & F_{-2,2,0} & F_{-1,2,0} & F_{0,2,0} & A & A & A \\ F_{-3,3,0} & F_{-2,3,0} & F_{-1,3,0} & A & A & A & A \end{bmatrix}$$

donde el primer subíndice i indica la coordenada x del pixel, la cual crece de izquierda a derecha, y el segundo j indica la coordenada y del pixel, creciendo de arriba hacia abajo. Los valores indicados como $F_{i,j,0}$ son desconocidos a priori.

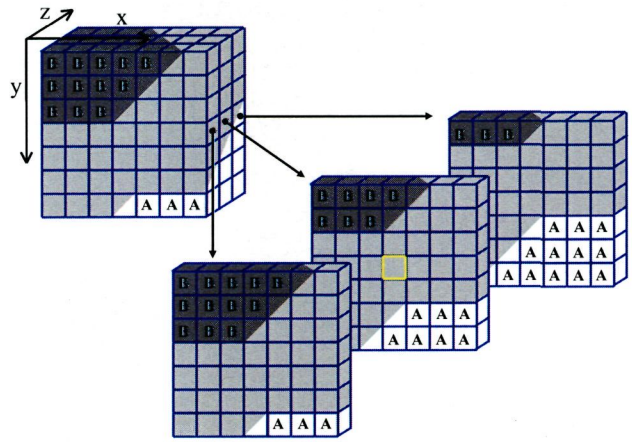


Figura 7.8: Los voxels de color gris claro son aquéllos por donde un plano tal que $F_y \geq F_x, F_z \geq 0$ puede pasar

Tomemos ahora una máscara de convolución para las derivadas parciales en y, H_y (ecuación 7.2), pero tomando el caso $\alpha = \beta = 0$ para mayor simplicidad. En este caso, tenemos que

$$H_y = \frac{1}{2h_y}\left[\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\right]$$

Si convolucionamos la imagen F con dicha máscara, obtendremos la imagen de parciales F_y tal que

$$F_y(i,j,k) = \frac{1}{2h_y}\left(F_{i,j+1,k} - F_{i,j-1,k}\right)$$

Observando la columna central de la imagen original, $F(0,j,0)$, vemos que es igual a B si $j < -2$ e igual a A si $j > 2$. Esto implica que la columna central de la imagen de parciales, $F_y(0,j,0)$, es igual a cero si $|j| > 3$. El resultado de realizar la suma de los 7 valores no nulos de la columna central de la imagen de parciales en y es el siguiente:

$$S_y = \sum_{j=-3}^{3} F_y(0,j,0) = \frac{A-B}{h_y}$$

el cual coincide con el salto de intensidad producido a ambos lados del borde, que era una de las incógnitas que queríamos obtener. Así damos pie al primer lema del capítulo.

Lema 7.1 *Sea $F_{i,j,k}(A,B,a,b,c,e)$ una imagen ideal con un contorno de primer orden que atraviesa el voxel central, tal que $b \geq a, c \geq 0$. La suma de los 7 valores no nulos de la columna central de la imagen de parciales en y es igual a*

$$S_y = \frac{A-B}{h_y} \tag{7.6}$$

donde h_y indica la longitud del voxel en la dirección y.

7.4 Cálculo de la orientación del contorno

Para el cálculo del salto de intensidad hemos usado exclusivamente la imagen de parciales en y. Ahora usaremos las otras dos imágenes de parciales, en x y en z, para obtener la orientación del plano del borde.

7.4.1 Derivadas parciales en la dirección x

Comencemos por la derivada en la dirección x, y para ello tomemos una máscara de convolución H_x donde $\alpha = \beta = 0$, cuya expresión es

$$H_x = \frac{1}{2h_x}\left[\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\right]$$

Al convolucionar la imagen F con esta máscara obtendremos la imagen de parciales F_x, tal que

$$F_x(i,j,k) = \frac{1}{2h_x}\left(F_{i+1,j,k} - F_{i-1,j,k}\right)$$

Esta imagen, al igual que ocurría con la imagen F_y, tendrá en su columna central, $F_x(0,j,0)$, hasta 7 valores no nulos. El resultado de sumar esos siete valores da lugar a la expresión:

$$S_x = \sum_{j=-3}^{3} F_x(0,j,0) = \frac{A-B}{h_x} + \frac{1}{2h_x}\left(\sum_{j=-3}^{1} F_{1,j,0} - \sum_{j=-1}^{3} F_{-1,j,0}\right) \tag{7.7}$$

Visualmente, la expresión es una suma ponderada de los voxels que se muestran en la figura 7.9. Algunos de esos voxels sabemos de antemano que tendrán valor A o B, pero otros (mostrados en color gris) dependerán de la orientación y la posición exacta del plano. Por ejemplo, si el plano fuese prácticamente horizontal, los únicos voxels que tendrían valor intermedio entre A y B serían los de la fila central, ya que serían los únicos voxels atravesados por el plano. Consideremos primero este caso, y luego estudiemos el resto de situaciones para diferentes orientaciones del plano.

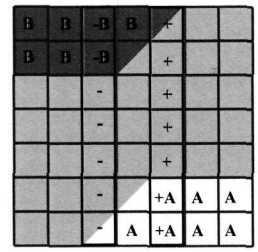


Figura 7.9: Voxels utilizados en la expresión para S_x (todos en la sección $z = 0$)

Caso horizontal

Supongamos que el plano del borde atraviesa sólamente los voxels de la fila central, es decir, los voxels $(-1,0,0)$, $(0,0,0)$ y $(1,0,0)$, como se muestra en la figura 7.10, donde

$$0 \le t - c_1 \le t \le t + c_1 \le t + 2c_1 \le t + 2c_1 + c_2 \le h_y$$

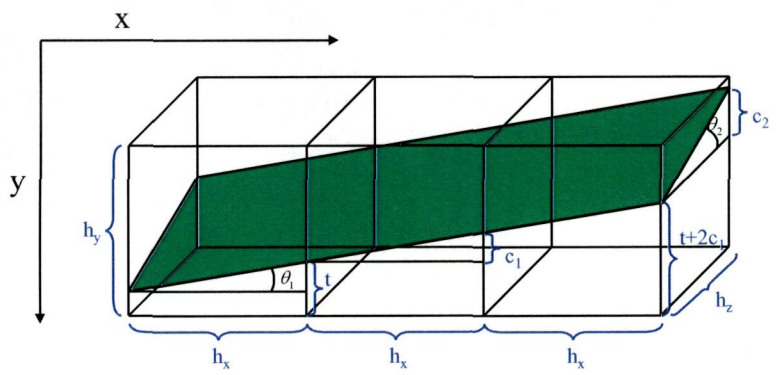


Figura 7.10: El plano del borde atraviesa sólamente los voxels de la fila central.

En este caso, los voxels situados por encima de esta fila tendrán valor B, y los de debajo valor A. Para poder evaluar la expresión de S_x faltaría calcular la intensidad de los voxels $(-1,0,0)$ y $(1,0,0)$, que según nuestra hipótesis de trabajo (ecuación 7.4) son

$$\begin{aligned} F_{-1,0,0} &= B + \frac{V_{-1,0,0}}{h_x h_y h_z}(A-B) \\ F_{1,0,0} &= B + \frac{V_{1,0,0}}{h_x h_y h_z}(A-B) \end{aligned}$$

donde $V_{-1,0,0}$ y $V_{1,0,0}$ son los volúmenes bajo el plano interior a cada voxel, cuyas expresiones son

$$\begin{aligned} V_{-1,0,0} &= \frac{h_x h_z}{2}(2t - c_1 + c_2) \\ V_{1,0,0} &= \frac{h_x h_z}{2}(2t + 3c_1 + c_2) \end{aligned}$$

Ahora ya podemos obtener la expresión para S_x:

$$\begin{aligned} S_x &= \frac{A-B}{h_x} + \frac{1}{2h_x}(3B + F_{1,0,0} + A - B - F_{-1,0,0} - 3A) = \\ &= \frac{1}{2h_x}(F_{1,0,0} - F_{-1,0,0}) = \frac{c_1}{h_x h_y}(A-B) \end{aligned}$$

Este resultado es muy interesante ya que, dividido por el valor obtenido para la suma de las parciales en y (ecuación 7.6) nos da exactamente la pendiente del plano en la dirección x, que era otra de las incógnitas que queríamos estimar. Esto es

$$\frac{S_x}{S_y} = \frac{c_1}{h_x}$$

Resto de casos

Faltaría ver que en todos los casos restantes se cumple esta misma propiedad. Eso implicaría calcular en cada uno las expresiones para el valor de intensidad de todos los voxels por donde pasa el plano. Afortunadamente,

hay una manera de demostrar la propiedad para todos los casos a la vez, en lugar de ir analizando caso a caso. Llamemos como hicimos antes $V_{i,j,k}$ al volumen interior al voxel (i,j,k) que cae bajo el plano. En el caso que el plano no atraviese el voxel en cuestión, $V_{i,j,k}$ valdrá 0 si el plano pasa por debajo del voxel, y $h_x h_y h_z$ si pasa por arriba. De esta forma tendremos que la intensidad de un voxel (i,j,k) vendrá dada por la expresión

$$F(i,j,k) = \frac{V_{i,j,k}}{h_x h_y h_z} A + \frac{h_x h_y h_z - V_{i,j,k}}{h_x h_y h_z} B = B + \frac{(A-B)}{h_x h_y h_z} V_{i,j,k} \tag{7.8}$$

Esto significa que la expresión de S_x (ecuación 7.7) vale

$$\begin{aligned} S_x &= \sum_{j=-3}^{3} F_x(0,j,0) = \frac{1}{2h_x}\left(\sum_{j=-3}^{3} F_{1,j,0} - \sum_{j=-3}^{3} F_{-1,j,0}\right) = \\ &= \frac{A-B}{2h_x^2 h_y h_z}\left(\sum_{j=-3}^{3} V_{1,j,0} - \sum_{j=-3}^{3} V_{-1,j,0}\right) = \frac{A-B}{2h_x^2 h_y h_z}(R-L) \end{aligned}$$

con lo cual vemos que lo importante no es el valor de los volúmenes $V_{i,j,k}$ individuales, sino el valor de la suma de ellos en una misma columna (R para la columna derecha y L para la izquierda).

Este factor que aparece en la expresión, $R-L$, es en realidad una diferencia de volúmenes, como se aprecia en la figura 7.11. Teniendo en cuenta que el plano atraviesa el voxel central $(0,0,0)$ y que sus pendientes en las direcciones x y z son menores que $\pi/4$ cada una, puede garantizarse que el plano no alcanzará nunca el techo del voxel $(1,-3,0)$ ni el suelo del voxel $(-1,3,0)$. Por lo tanto, la diferencia $R-L$ será siempre un prisma como el de la figura, cuyo volumen vendrá dado por la expresión:

$$R - L = 2c_1 h_x h_z$$

independientemente de los voxels que sean atravesados por el plano. De hecho, puede apreciarse como la expresión que sale es independiente de la ubicación exacta del plano, t, y de la pendiente en la dirección z, c_2.

Con esto, la expresión para S_x en cualquiera de los casos queda como:

$$S_x = \frac{A-B}{2h_x^2 h_y h_z}(2c_1 h_x h_z) = \frac{c_1}{h_x h_y}(A-B)$$

que es la misma que obtuvimos en el caso horizontal

7.4.2 Derivadas parciales en la dirección z

El razonamiento para deducir la segunda pendiente del plano en la dirección z es exactamente igual que para la dirección x. Partimos de una máscara de convolución similar, H_z, donde $\alpha = \beta = 0$, cuya expresión es

$$H_z = \frac{1}{2h_z}\left[\begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}\right]$$

Al convolucionar la imagen F con esta máscara obtendremos la imagen de parciales F_z, tal que

$$F_z(i,j,k) = \frac{1}{2h_z}(F_{i,j,k+1} - F_{i,j,k-1})$$

El resultado de sumar los siete valores no nulos de su columna central es el siguiente:

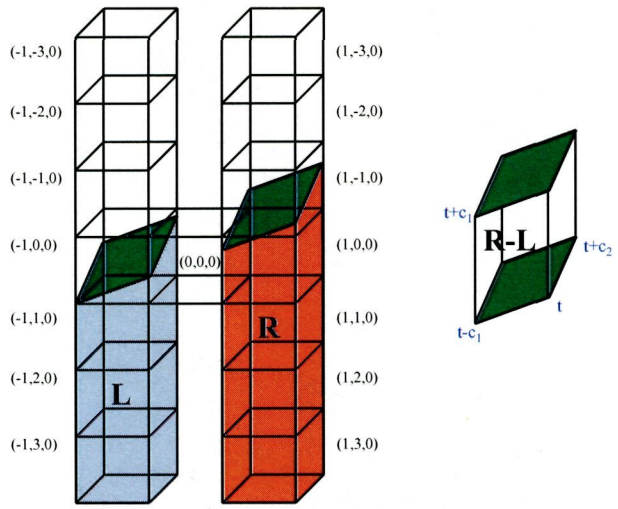


Figura 7.11: La suma de las parciales en x de la columna central es proporcional a la diferencia de volúmenes $R - L$

$$\begin{aligned} S_z &= \sum_{j=-3}^{3} F_z(0,j,0) = \frac{1}{2h_z}\left(\sum_{j=-3}^{3} F_{0,j,1} - \sum_{j=-3}^{3} F_{0,j,-1}\right) = \\ &= \frac{A-B}{2h_x h_y h_z^2}\left(\sum_{j=-3}^{3} V_{0,j,1} - \sum_{j=-3}^{3} V_{0,j,-1}\right) = \frac{A-B}{2h_x h_y h_z^2}(R-L) \end{aligned}$$

donde ahora R representa el volumen bajo el plano interior a la columna $z = 1$, y L el volumen bajo el plano interior a la columna $z = -1$. La expresión para la resta $R - L$ sale similar

$$R - L = 2c_2 h_x h_z$$

con lo cual finalmente la expresión para S_z queda como sigue:

$$S_z = \frac{c_2}{h_y h_z}(A - B)$$

Dividiendo dicha expresión por S_y obtenemos la pendiente del plano en la dirección z. Con todo esto damos pie al siguiente lema:

Lema 7.2 *Sea $F_{i,j,k}(A, B, a, b, c, e)$ una imagen ideal con un contorno de primer orden que atraviesa el voxel central, tal que $b \geq a, c \geq 0$. La suma de los 7 valores no nulos de la columna central de las imágenes de parciales en x y en z es igual a*

$$\begin{aligned} S_x &= \frac{c_1}{h_x h_y}(A - B) \\ S_z &= \frac{c_2}{h_y h_z}(A - B) \end{aligned}$$

donde (h_x, h_y, h_z) indican las dimensiones del voxel, y donde se cumple que $a = c_1/h_x$ y $c = c_2/h_z$.

7.4.3 Cálculo del vector normal

Con los dos lemas que hemos expuesto ya podemos dar una expresión para el vector normal al plano del borde con exactitud. La situación es la que se muestra en la figura 7.12. Buscamos un vector normal cuyo módulo represente el salto de intensidad producido a ambos lados del plano, y que su orientación indique la dirección de crecimiento de la intensidad en la imagen (por tanto en el dibujo, el vector se muestra invertido, ya que $A > B$, siendo A la intensidad bajo el plano y B sobre él).

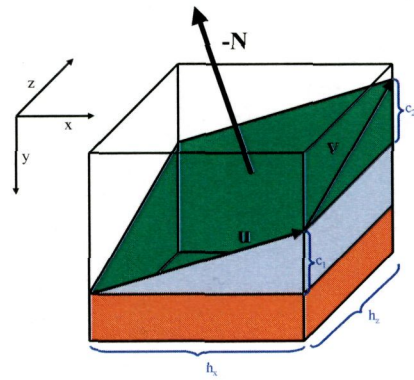


Figura 7.12: Vector normal al plano del borde

Atendiendo a dicha figura, la dirección del vector normal ha de ser igual a la del producto vectorial $v \times u$, donde $v = [0, -c_2, h_z]$ y $u = [h_x, -c_1, 0]$. Dicho producto es igual a

$$v \times u = \begin{bmatrix} c_1 h_z & h_x h_z & c_2 h_x \end{bmatrix}$$

El módulo del vector normal debe ser igual a $S_y = (A - B)/h_y$. Así que podemos suponer que

$$N = \frac{1}{K} \begin{bmatrix} c_1 h_z & h_x h_z & c_2 h_x \end{bmatrix}$$

donde K es un factor de escala para que cumpla que $\|N\| = S_y$. De esta forma, se deduce el valor de K como sigue

$$K = \frac{1}{S_y}\sqrt{c_1^2 h_z^2 + h_x^2 h_z^2 + c_2^2 h_x^2} = \frac{h_x h_z}{S_y}\sqrt{\left(\frac{S_x}{S_y}\right)^2 + 1 + \left(\frac{S_z}{S_y}\right)^2} = \frac{h_x h_z}{S_y^2}\sqrt{S_x^2 + S_y^2 + S_z^2}$$

donde se han usado las equivalencias $S_x/S_y = c_1/h_x$ y $S_z/S_y = c_2/h_z$ en uno de los pasos. Con esto ya llegamos a la expresión final para el vector normal, que es la siguiente:

$$N = \frac{S_y^2}{h_x h_z \sqrt{S_x^2 + S_y^2 + S_z^2}} \begin{bmatrix} h_x h_z \frac{S_x}{S_y} & h_x h_z & h_x h_z \frac{S_z}{S_y} \end{bmatrix} = \frac{S_y}{\sqrt{S_x^2 + S_y^2 + S_z^2}} \begin{bmatrix} S_x & S_y & S_z \end{bmatrix}$$

Este resultado se resume en el siguiente lema:

Lema 7.3 *Sea $F_{i,j,k}(A, B, a, b, c, e)$ una imagen ideal con un contorno de primer orden que atraviesa el voxel central, tal que $b \geq a, c \geq 0$. Sean S_x, S_y y S_z las sumas de los 7 valores no nulos de la columna central de las imágenes de parciales en x, y y z respectivamente. El vector normal al contorno, N, cuyo módulo indica la diferencia de intensidad existente a ambos lados del borde, viene dado por la expresión*

$$N = \frac{S_y}{\sqrt{S_x^2 + S_y^2 + S_z^2}} \begin{bmatrix} S_x & S_y & S_z \end{bmatrix}$$

7.5 Elección del pixel borde correcto

De todos los voxels de la columna central, ¿cómo identificar cuál de los voxels es el más cercano al plano del borde, para etiquetarlo como tal y realizar los cálculos centrados en él?. Hay que darse cuenta que el plano del borde puede pasar por un sólo voxel de la columna, por dos voxels, e incluso por tres voxels, como puede apreciarse en la figura 7.13. Veremos que en los tres casos, aquel voxel con derivada parcial en y mayor es el voxel más indicado de la columna para ser etiquetado como perteneciente al borde.

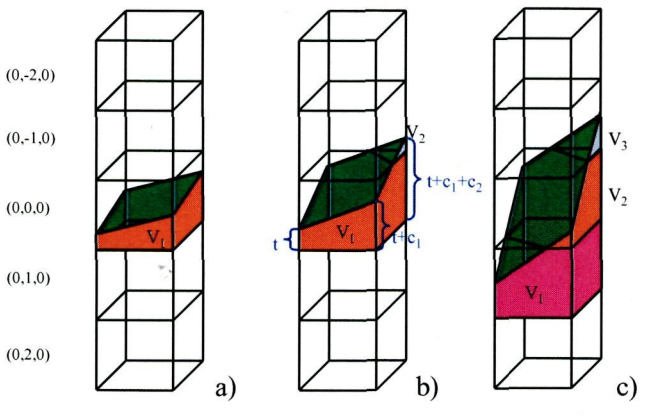


Figura 7.13: Un plano con componente y máxima puede cruzar 1, 2 ó 3 voxels de una misma columna

Antes de empezar a cada caso, llamemos V_1^*, V_2^* y V_3^* a los volúmenes relativos de los volúmenes V_1, V_2 y V_3 que aparecen en la figura 7.13. Es decir

$$V_n^* = \frac{V_n}{h_x h_y h_z} \quad \forall n \in \{1, 2, 3\}$$

De esta forma sabemos que $0 < V_n^* < 1 \ \forall n$.

Caso a: el contorno atraviesa sólo el voxel central En este primer caso, las intensidades y parciales de los voxels son las siguientes:

$$F(0, j, 0) = \begin{bmatrix} B \\ B \\ B + (A - B) V_1^* \\ A \\ A \end{bmatrix}; \quad F_y(0, j, 0) = \frac{1}{2h_y} \begin{bmatrix} 0 \\ (A - B) V_1^* \\ A - B \\ (A - B)(1 - V_1^*) \\ 0 \end{bmatrix}$$

Por tanto queda claro que la parcial en y del voxel central es la de mayor valor en la columna.

Caso b: el contorno atraviesa dos voxels en la columna Veamos ahora los datos para este caso:

$$F(0,j,0) = \begin{bmatrix} B \\ B+(A-B)V_2^* \\ B+(A-B)V_1^* \\ A \\ A \end{bmatrix}; \quad F_y(0,j,0) = \frac{1}{2h_y} \begin{bmatrix} (A-B)V_2^* \\ (A-B)V_1^* \\ (A-B)(1-V_2^*) \\ (A-B)(1-V_1^*) \\ 0 \end{bmatrix}$$

Atendiendo a la figura 7.13b, se cumple siempre que $V_1^* > V_2^*$ con lo cual $F_y(0,-2,0) < F_y(0,-1,0)$. Con esto descartamos el voxel $(0,-2,0)$ como valor máximo. Del mismo modo $1-V_1^* < 1-V_2^*$, con lo cual $F_y(0,1,0) < F_y(0,0,0)$, y descartamos también el voxel $(0,1,0)$. Nos quedan los dos voxels centrales, $(0,0,0)$ y $(0,-1,0)$, para deducir cuál tiene la mayor parcial. Para que sea el $(0,0,0)$, deberá cumplirse que $1-V_2^* \geq V_1^*$, es decir, $V_1^* + V_2^* \leq 1$. Este último volumen $V_1^* + V_2^*$ tiene la expresión siguiente:

$$V = V_1^* + V_2^* = \frac{1}{h_y}\left(t + \frac{c_1+c_2}{2}\right)$$

De aquí se deduce que la inecuación se convierte en

$$h_y \geq t + \frac{c_1+c_2}{2}$$

La parte derecha de la expresión se corresponde con la distancia en vertical del punto central del plano del borde hasta suelo del voxel $(0,0,0)$. Como h_y mide la altura del voxel, concluimos que el voxel de parcial máxima será aquél que incluya al punto central del plano del borde (y que por lo tanto incluirá más superficie de borde que el otro voxel).

Caso c: el contorno atraviesa tres voxels en la columna Los datos para este último caso son:

$$F(0,j,0) = \begin{bmatrix} B \\ B+(A-B)V_3^* \\ B+(A-B)V_2^* \\ B+(A-B)V_1^* \\ A \end{bmatrix}; \quad F_y(0,j,0) = \frac{1}{2h_y} \begin{bmatrix} (A-B)V_3^* \\ (A-B)V_2^* \\ (A-B)(V_1^*-V_3^*) \\ (A-B)(1-V_2^*) \\ (A-B)(1-V_1^*) \end{bmatrix}$$

Atendiendo a la figura 7.13c, se cumple siempre que $V_2^* \geq V_3^*$, con lo cual descartamos el voxel $(0,-2,0)$. Tambien se cumple que $1-V_1^* \leq 1-V_2^*$, con lo cual descartamos el voxel $(0,2,0)$. Otra condición que se cumple es que $V_1^* \geq V_2^* + V_3^*$, con lo cual $V_1 - V_3 \geq V_2$, con lo que descartamos el voxel $(0,-1,0)$. Finalmente, un razonamiento análogo nos lleva a que $1-V_3^* \geq (1-V_2^*)+(1-V_1^*)$, de donde se deduce que $V_1^* - V_3^* \geq 1-V_2^*$, con lo cual descartamos también el voxel $(0,1,0)$. De todo ello deducimos que el voxel con parcial mayor es el $(0,0,0)$.

Una vez analizados los tres casos, podemos observar cómo el voxel que contenga mayor superficie del borde en su interior será el que mayor derivada parcial en y tenga. Esta conclusión se resume en el siguiente lema:

Lema 7.4 *Sea $F_{i,j,k}(A,B,a,b,c,e)$ una imagen ideal con un contorno de primer orden que atraviesa el voxel central, tal que $b \geq a, c \geq 0$. Se cumple que, para cada columna, el pixel (i,j,k) por el que pasa un área de contorno más largo es el que cumple que*

$$0 \leq F_y(i,j-2,k) < F_y(i,j-1,k) < F_y(i,j,k) > F_y(i,j+1,k) > F_y(i,j+2,k) \geq 0$$

Usando este lema, en nuestra imagen F ideal con la que estamos trabajando, obtendríamos una imagen de bordes con un único voxel borde por columna. Dicho de otra forma, los bordes obtenidos a partir de la imagen

serían siempres superficies de un solo voxel de grosor. Esta propiedad es muy interesante porque existen algunos detectores de borde [MON91, LOH98] que producen superficies de borde de grosor mayor en algunas zonas, con lo cual son menos precisos a la hora de estimar la localización exacta del contorno. De hecho, necesitan etapas posteriores de "adelgazamiento" (edge thinning) para reducir el número de voxels borde [LOH98].

7.6 Cálculo de la posición exacta del contorno

Veamos ahora cómo calcular la posición exacta del plano del borde en el interior del voxel. Y para ello vamos a realizar el siguiente proceso. Primero habrá que detectar el valor máximo de la derivada parcial en y de la columna, tal y como se vio en la sección anterior. Podemos suponer que el voxel elegido tenga por coordenadas $(0,0,0)$. Calculemos ahora cuál es, de sus dos vecinos en vertical, el que tenga mayor derivada parcial. Si es el $(0,-1,0)$, (el vecino de arriba), la expresión que evaluaremos será

$$R_y^- = 3F_y(0,-3,0) + F_y(0,-2,0) + F_y(0,-1,0) - F_y(0,0,0) - F_y(0,1,0) - 3F_y(0,2,0)$$

En caso que el vecino con mayor parcial sea el $(0,1,0)$, (el vecino de abajo), la expresión será igual pero desplazada una unidad hacia abajo

$$R_y^+ = 3F_y(0,-2,0) + F_y(0,-1,0) + F_y(0,0,0) - F_y(0,1,0) - F_y(0,2,0) - 3F_y(0,3,0)$$

Analizemos estas expresiones en los tres casos de la sección anterior.

Caso a: el contorno atraviesa sólo el voxel central Cuando el plano del borde sólo cubre un voxel, la parcial máxima siempre está en el $(0,0,0)$. Si el vecino con mayor parcial es el de arriba obtenemos

$$R_y^- = \frac{A-B}{h_y}(V_1^* - 1)$$

Teniendo en cuenta que

$$V_1^* = \frac{1}{h_y}\left(t + \frac{c_1 + c_2}{2}\right)$$

nos queda que

$$R_y^- = \frac{A-B}{h_y^2}\left[\left(t + \frac{c_1+c_2}{2}\right) - h_y\right] = \frac{A-B}{h_y}\frac{d_1}{h_y} = S_y\frac{d_1}{h_y}$$

donde d_1 es la distancia en vertical entre el centro geométrico del plano del borde y el punto central de la cara superior del voxel central, como se ve en la figura 7.14.

Supongamos ahora que el vecino mayor es el de abajo. En ese caso, aplicaríamos la segunda expresión para R_y, con la que obtendríamos

$$R_y^+ = \frac{A-B}{h_y}V_1^* = \frac{A-B}{h_y^2}\left(t + \frac{c_1+c_2}{2}\right) = \frac{A-B}{h_y}\frac{d_2}{h_y} = S_y\frac{d_2}{h_y}$$

donde ahora d_2 es la distancia en vertical entre el centro geométrico del plano del borde y el punto central de la cara inferior del voxel central. Vemos que en ambos casos la distancia va siempre medida con respecto al punto central de la cara compartida por los dos voxels de la columna con derivada parcial máxima.

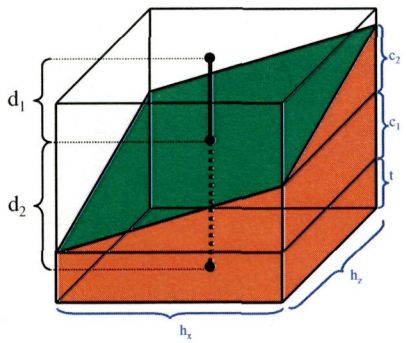


Figura 7.14: El contorno atraviesa sólo el voxel central

Caso b: el contorno atraviesa dos voxels en la columna En este caso, las dos parciales máximas siempre son las de los dos voxels que comparten el borde. Supongamos que la situación es la mostrada en la figura 7.13b, donde el plano atraviesa el voxel vecino superior. En este caso, es fácil comprobar que si $F_y(0,-1,0) > F_y(0,1,0)$ entonces se cumple que $V_1^* > 1 - V_1^*$, lo que a su vez supone que

$$t + \frac{c_1 + c_2}{2} > \frac{1}{2} h_y$$

Es decir, que el centro geométrico del plano del borde está más próximo al voxel vecino superior que al inferior, cosa que ocurre. Por lo tanto, la expresión en este caso para el desplazamiento es la siguiente

$$R_y^- = \frac{A - B}{h_y}(V - 1)$$

donde $V = V_1^* + V_2^*$. Como la expresión para V es la misma que se desarrolló para este mismo caso en la sección anterior, obtenemos que

$$R_y^- = \frac{A - B}{h_y^2}\left[\left(t + \frac{c_1 + c_2}{2}\right) - h_y\right] = \frac{A - B}{h_y}\frac{d_1}{h_y} = S_y \frac{d_1}{h_y}$$

donde d_1 vuelve a ser la distancia en vertical entre el centro geométrico del plano del borde y el punto central de la cara superior del voxel central, como se ve en la figura 7.15.

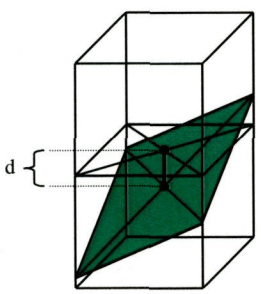


Figura 7.15: El contorno atraviesa el voxel central y el superior

Si el plano hubiese atravesado el voxel inferior, se usaría la expresión R_y^+ y obtendríamos un resultado similar, donde la distancia vendría medida con respecto a la cara inferior del voxel central. En función de en qué caso estemos, la distancia será positiva o negativa.

Caso c: el contorno atraviesa tres voxels en la columna En este último caso, que aparece en la figura 7.13c, el razonamiento es análogo. Supongamos que el vecino en vertical con mayor valor de parcial es el de arriba. En ese caso, la expresión para el desplazamiento es la siguiente:

$$R_y^- = \frac{A-B}{h_y}(V-2)$$

donde $V = V_1^* + V_2^* + V_3^*$. En este caso

$$R_y^- = \frac{A-B}{h_y^2}\left[\left(t + \frac{c_1+c_2}{2}\right) - 2h_y\right] = \frac{A-B}{h_y}\frac{d_1}{h_y} = S_y\frac{d_1}{h_y}$$

donde d_1 vuelve a ser la distancia en vertical entre el centro geométrico del plano del borde y el punto central de la cara superior del voxel central, como se ve en la figura 7.16. Obsérvese cómo en este caso el parámetro t va medido desde el suelo del voxel $(0,1,0)$.

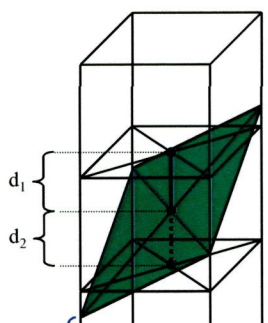


Figura 7.16: El contorno atraviesa tres voxels en la columna

Si el voxel vecino de mayor parcial fuese el inferior, la expresión saldría

$$R_y^+ = \frac{A-B}{h_y}(V-1) = \frac{A-B}{h_y^2}\left(t + \frac{c_1+c_2}{2} - h_y\right) = \frac{A-B}{h_y}\frac{d_2}{h_y} = S_y\frac{d_2}{h_y}$$

donde ahora d_2 es la distancia en vertical entre el centro geométrico del plano del borde y el punto central de la cara inferior del voxel central, como se ve en la figura 7.16.

Medición de la distancia desde el centro del pixel Para que la distancia d no esté medida unas veces con respecto a la cara inferior del voxel central, y otras con respecto a la cara superior, llamemos p a una nueva cantidad que represente la diferencia de altura entre el plano y el centro geométrico del voxel, como se ve en la figura 7.17. Así, la expresión para p será

$$p = \begin{cases} -h_y/2 - d & \text{si } F_y(0,-1,0) > F_y(0,1,0) \\ h_y/2 - d & \text{si } F_y(0,-1,0) < F_y(0,1,0) \end{cases}$$

siendo $0 < p < h/2$ cuando el contorno pasa por debajo del centro del pixel, y $-h/2 < p < 0$ cuando pasa por encima.

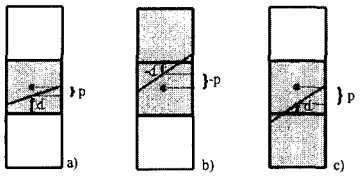


Figura 7.17: La distancia al contorno va medida en vertical con respecto al centro geométrico del voxel.

Con todo lo anterior ya podemos enunciar el próximo lema:

Lema 7.5 *Sea $F_{i,j,k}(A, B, a, b, c, e)$ una imagen ideal con un contorno de primer orden que atraviesa el voxel central, tal que $b \geq a, c \geq 0$, y que cumple que $F_y(0,-1) < F_y(0,0) > F_y(0,1)$. Sea R_y la expresión siguiente*

$$\begin{aligned} R_y \;=\; & 3F_y(0,m-3,0) + F_y(0,m-2,0) + F_y(0,m-1,0) - \\ & -F_y(0,m,0) - F_y(0,m+1,0) - 3F_y(0,m+2,0) \end{aligned} \tag{7.9}$$

donde

$$m = \begin{cases} 0 & \text{si } F_y(0,-1,0) \geq F_y(0,1,0) \\ 1 & \text{si } F_y(0,-1,0) < F_y(0,1,0) \end{cases}$$

El desplazamiento exacto del contorno dentro del voxel viene dado por la ecuación

$$\frac{p}{h_y} = \frac{2m-1}{2} - \frac{R_y}{S_y}$$

donde S_y es la suma de los 7 valores no nulos de la columna central de la imagen de parciales en y, h_y es la altura del voxel, y p es la diferencia de altura entre el contorno y el centro geométrico del voxel $(0,0,0)$.

7.7 Generalización para el resto de octantes

Todo el estudio anterior se ha realizado considerando contornos planos que seguían la expresión

$$ax + by + cz + e = 0$$

donde se cumplía que $b \geq a, c \geq 0$. Además, se consideraba el caso donde la intensidad bajo el plano, A, era mayor que la intensidad por encima, B. Es decir, al calcular las tres imágenes de parciales, suponíamos que

$$F_y > F_x, F_z \geq 0$$

en el voxel seleccionado. Esta última condición realmente es sólo válida para el caso en el que $h_x = h_y = h_z$. En caso contrario, la condición que debería cumplirse es que

$$F_y h_y > F_x h_x, F_z h_z \geq 0$$

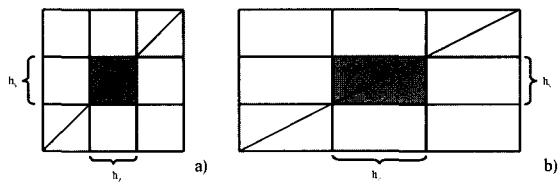


Figura 7.18: a) $h_y = h_z$; b) $2h_y = h_z$

Esto puede apreciarse mejor en la figura 7.18, donde se muestran dos situaciones: en la primera $h_y = h_z$, y en la segunda $h_z = 2h_y$. En ambos casos se muestra la pendiente límite a partir de la cual cambiaríamos de casos, la cual se corresponde cuando $F_y h_y = F_z h_z$.

Por tanto, tratemos primero el caso general en el que

$$|F_y|\, h_y > |F_x|\, h_x, |F_z|\, h_z$$

Este caso general engloba 8 casos diferentes, en función del signo de las tres parciales. Es fácil ver que generalizando todos los lemas anteriores del presente capítulo podemos llegar al siguiente lema para estos 8 casos conjuntamente:

Lema 7.6 *Sea $F_{i,j,k}(A, B, a, b, c, e)$ una imagen ideal con un contorno de primer orden que atraviesa el voxel central, y que cumple que*

$$\begin{aligned} |F_y(0,0,0)|\, h_y &> |F_x(0,0,0)|\, h_x, |F_y(0,0,0)|\, h_z \\ |F_y(0,1,0)| &< |F_y(0,0,0)| > |F_y(0,-1,0)| \end{aligned}$$

El vector normal al contorno, N, cuyo módulo indica la diferencia de intensidad existente a ambos lados del borde, viene dado por la expresión

$$N = \frac{|S_y|}{\sqrt{S_x^2 + S_y^2 + S_z^2}} \begin{bmatrix} S_x & S_y & S_z \end{bmatrix}$$

y su posición vertical relativa dentro del voxel viene dada por la expresión

$$\frac{p}{h_y} = \frac{2m-1}{2} - \frac{R_y}{S_y}$$

donde

$$\begin{aligned} S_x &= \sum_{j=-3}^{3} F_x(0,j,0) \\ S_y &= \sum_{j=-3}^{3} F_y(0,j,0) \\ S_z &= \sum_{j=-3}^{3} F_z(0,j,0) \\ R_y &= 3F_y(0,m-3,0) + F_y(0,m-2,0) + F_y(0,m-1,0) - \\ &\quad -F_y(0,m,0) - F_y(0,m+1,0) - 3F_y(0,m+2,0) \\ m &= \begin{cases} 0 & \text{si } |F_y(0,-1,0)| \geq |F_y(0,1,0)| \\ 1 & \text{si } |F_y(0,-1,0)| < |F_y(0,1,0)| \end{cases} \end{aligned}$$

Quedarían por tratar otros 16 casos, 8 cuando la mayor parcial en valor absoluto sea F_x y otros 8 cuando sea F_z. Los lemas para estas dos situaciones serían análogos al lema que acabamos de exponer, cambiando unas variables por otras, y realizando las sumatorias en la dirección de la variable cuya parcial sea máxima.

7.8 Simplificación a un entorno más reducido

Observando las expresiones usadas en el lema anterior, basadas en los valores de las imágenes de parciales, vemos que usan información de voxels muy alejados al voxel central. Esta característica no suele ser muy aconsejable, ya que tratamos de obtener información local a cada voxel, y cuanto más extensa sea la vecindad de voxels que usemos para obtener los parámetros del contorno de un voxel concreto, menos preciso serán los valores estimados. Veamos otra alternativa para obtener los mismos valores, pero sin tener que alejarnos tanto del voxel central.

7.8.1 Suma de las parciales en y

Realmente lo que hacemos al sumar la columna de parciales en y es realizar un promedio de los voxels con valor A y de los voxels con valor B y hacer la resta. La diferencia entre usar alfa y beta nulos o no en las máscaras de convolución es que el promedio se realiza entre un número mayor de voxels. Por ejemplo, para el caso alfa y beta nulos se usan dos voxels para cada valor:

$$\begin{aligned} A &\simeq F_{0,3,0} + F_{0,4,0} \\ B &\simeq F_{0,-4,0} + F_{0,-3,0} \end{aligned}$$

Para el caso alfa y beta no nulos el resultado es bastante más complejo, ya que la expresión de S_y es una suma enorme de voxels, donde los sumandos positivos estiman el valor de A y los negativos el de B. Exactamente, tanto para A como para B se están usando 22 voxels. Además, son voxels un poco alejados al voxel central. Por ejemplo, para el cálculo de A se están usando voxels como el $(1, 6, 1)$, que están a más de 6 voxels de distancia del voxel central. Puede ser demasiada distancia en algunos casos.

Por ello, una alternativa al cálculo del salto de intensidad puede ser la siguiente. Ya que lo que se busca es estimar A y B, una estimación sencilla usando una ventana de 3x3x3, y atendiendo a los voxels cuya intensidad conocemos de antemano que será A o B en cualquier caso, según se mostraba en la figura 7.8, puede ser

$$\begin{aligned} A &\simeq F_{1,1,1} \\ B &\simeq F_{-1,-1,-1} \end{aligned}$$

Quizás esta fórmula no es muy adecuada para una imagen digitalizada, pues se está usando únicamente el valor de 1 voxel para estimar cada intensidad. Mejor sería usar una máscara 5x3x5, lo cual daría un resultado más suave. En ese caso, la estimación sería

$$\begin{aligned} A &\simeq \frac{1}{4}\left(F_{1,2,0} + F_{0,2,1} + F_{1,1,1} + F_{1,2,1}\right) \\ B &\simeq \frac{1}{4}\left(F_{-1,-2,0} + F_{0,-2,-1} + F_{-1,-1,-1} + F_{-1,-2,-1}\right) \end{aligned}$$

Vemos que ahora se usa una información más cercana al voxel central, ya que el voxel más alejado está a distancia <2,1,2>. Y además es una estimación mejor con respecto al ruido, ya que estamos promediando entre 4 voxels diferentes para cada valor. Por lo tanto, calculando A y B con la expresión anterior, nuestra solución para S_y será:

$$S_y = \frac{A - B}{h_y}$$

7.8.2 Suma de las parciales en x y z

Para la suma de las parciales en x estamos usando la información de las columnas izquierda y derecha de la central, tomando 7 voxels en cada una. Sin embargo, los voxels importantes son los cinco de abajo en la columna de la izquierda, y los cinco de arriba en la derecha, como se vio en la figura 7.9, pues son los sitios posibles por donde el plano del borde puede pasar. Los otros dos voxels de cada columna son simplemente estimaciones para los valores A y B. Podemos usar para ellos las estimaciones que obtuvimos antes con una ventana 5x3x5, con lo cual la expresión para S_x quedaría:

$$S_x = \frac{A-B}{h_x} + \frac{1}{2h_x}\left(\sum_{j=-3}^{1} F_{1,j,0} - \sum_{j=-1}^{3} F_{-1,j,0}\right)$$

En esta expresión, los voxels más lejanos se encuentran a distancia <1,3,0> del voxel central, es decir, fuera de la ventana 5x3x5, pero a una distancia menor que las diagonales de dicha ventana.

Para las parciales en z se opera de la misma manera, pero usando las columnas de detrás y de delante del voxel central:

$$S_z = \frac{A-B}{h_z} + \frac{1}{2h_z}\left(\sum_{j=-3}^{1} F_{0,j,1} - \sum_{j=-1}^{3} F_{0,j,-1}\right)$$

De nuevo las distancias al voxel más lejano empleado en la fórmula coinciden con las de las parciales en x.

7.8.3 Desplazamiento del borde

En la expresión del desplazamiento (ecuación 7.9) se usaban 6 voxels en la columna central. Son por lo tanto voxels cercanos. Pero el problema radica en que los dos voxels más alejados llevan peso 3, es decir, tienen el triple de importancia que los voxels más cercanos al central, y esto no es muy conveniente. Una forma alternativa de estimar el desplazamiento es la siguiente: como ya se ha visto anteriormente, la intensidad de un voxel (i,j,k) viene dada por la expresión

$$F(i,j,k) = B + \frac{(A-B)}{h_x h_y h_z} V_{i,j,k}$$

donde $V_{i,j,k}$ es el volumen interior al voxel (i,j,k) que cae bajo el plano. Por lo tanto, podemos despejar el valor del volumen $V_{i,j,k}$ a partir de la intensidad de un voxel de la forma

$$V_{i,j,k} = h_x h_y h_z \frac{F_{i,j,k} - B}{A-B}$$

Si lo hacemos así para el voxel central y sus dos vecinos en vertical, tendremos el volumen de voxel que cae bajo el plano del borde. Esta expresión además funciona tanto para los casos en los que el borde sólo pasaba por un voxels como para los que pasaba por dos o tres, ya que si la intensidad del voxel es A, entonces el volumen $V_{i,j,k}$ coincide con el volumen del voxel. Y si la intensidad es B, entonces el volumen es cero.

Si ahora llamamos V a la suma de los tres volúmenes

$$\begin{aligned} V &= V_{0,-1,0} + V_{0,0,0} + V_{0,1,0} = \\ &= \frac{h_x h_y h_z}{A-B}\left(F_{0,-1,0} + F_{0,0,0} + F_{0,1,0} - 3B\right) \end{aligned}$$

Pero este volumen sabemos que es igual a $h_x h_z d$, donde d es la distancia en vertical desde el centro de la cara inferior del voxel $(0,-1,0)$, como se ve en la figura 7.19. Con lo cual, la distancia que buscamos es

$$d = \frac{h_y}{A-B}\left(F_{0,-1,0} + F_{0,0,0} + F_{0,1,0} - 3B\right)$$

Vemos que la información más lejana que hemos usado (aparte de la usada para estimar A y B), es simplemente de un voxel.

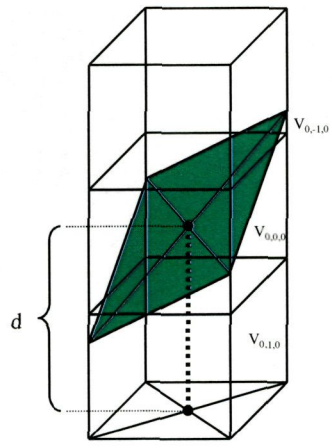


Figura 7.19: La distancia d va medida desde la cara inferior del voxel $(0, 1, 0)$

Para mayor claridad podemos considerar que la distancia fuese medida desde el centro geométrico del voxel central, en lugar de hacerlo desde la cara inferior del voxel vecino. Además, sería más claro que el signo de d coincidiera con el signo de la dirección en la que nos estamos moviendo (es decir, en este caso, que $d > 0$ indique que nos movemos hacia abajo, en la dirección de crecimiento de y). Por lo tanto, una expresión más correcta para la distancia sería

$$\begin{aligned} d &= \frac{3}{2}h_y - \frac{h_y}{A-B}\left(F_{0,-1,0} + F_{0,0,0} + F_{0,1,0} - 3B\right) = \\ &= \frac{3(A+B) - 2\left(F_{0,-1,0} + F_{0,0,0} + F_{0,1,0}\right)}{2S_y} \end{aligned}$$

7.8.4 Lema final

Con estos nuevas expresiones podemos llegar a otra versión del lema 7.6 anterior, que no use información de voxels muy alejados del voxel central:

Lema 7.7 *Sea $F_{i,j,k}(A, B, a, b, c, e)$ una imagen ideal con un contorno de primer orden que atraviesa el voxel central, y que cumple que*

$$\begin{aligned} |F_y(0,0,0)|\, h_y &> |F_x(0,0,0)|\, h_x, |F_y(0,0,0)|\, h_z \\ |F_y(0,1,0)| &< |F_y(0,0,0)| > |F_y(0,-1,0)| \end{aligned}$$

El vector normal al contorno, N, cuyo módulo indica la diferencia de intensidad existente a ambos lados del borde, viene dado por la expresión

$$N = \frac{|S_y|}{\sqrt{S_x^2 + S_y^2 + S_z^2}} \begin{bmatrix} S_x & S_y & S_z \end{bmatrix}$$

y su posición vertical relativa dentro del voxel viene dada por la expresión

$$d = \frac{3(A+B) - 2\left(F_{0,-1,0} + F_{0,0,0} + F_{0,1,0}\right)}{2S_y}$$

donde

$$\begin{aligned}
S_x &= \frac{A-B}{h_x} + \frac{1}{2h_x}\left(\sum_{j=-3}^{1} F_{1,j,0} - \sum_{j=-1}^{3} F_{-1,j,0}\right) \\
S_y &= \frac{A-B}{h_y} \\
S_z &= \frac{A-B}{h_z} + \frac{1}{2h_z}\left(\sum_{j=-3}^{1} F_{0,j,1} - \sum_{j=-1}^{3} F_{0,j,-1}\right) \\
A &= \frac{1}{4}\left(F_{1,2,0} + F_{0,2,1} + F_{1,1,1} + F_{1,2,1}\right) \\
B &= \frac{1}{4}\left(F_{-1,-2,0} + F_{0,-2,-1} + F_{-1,-1,-1} + F_{-1,-2,-1}\right)
\end{aligned}$$

7.9 Algoritmo para la localización de contornos

A partir del lema anterior ya podemos desarrollar un algoritmo para tratar de detectar la posición y orientación de los contornos en una imagen 3D. En primer lugar, habrá que detectar qué voxels son etiquetados como borde. Para ello, obtendremos las tres imágenes de parciales, F_x, F_y y F_z, y nos centraremos en aquellos voxels (i,j,k) donde

$$F_x^2(i,j,k) + F_y^2(i,j,k) + F_z^2(i,j,k) > \delta^2$$

donde δ es un umbral que habrá que prefijar. Aunque ya se ha comentado que esta expresión para el módulo del gradiente no es exacta, sí que puede usarse como aproximación para saber si existe un salto de intensidad grande en las cercanías del voxel.

Pero para catalogar un voxel como borde no bastará que cumpla esta condición. Además estamos interesados en localizar aquéllos voxels donde el valor absoluto de la derivada parcial crezca, llegue a un máximo, y vuelva a decrecer. Por lo tanto, la segunda condición que ha de cumplir un voxel para ser borde es que

$$|F_y(i,j-2,k)| < |F_y(i,j-1,k)| < |F_y(i,j,k)| > |F_y(i,j+1,k)| > |F_y(i,j+2,k)|$$

en el caso en que $|F_y(i,j,k)|\,h_y > |F_x(i,j,k)|\,h_x, |F_z(i,j,k)|\,h_z$, o

$$|F_x(i-2,j,k)| < |F_x(i-1,j,k)| < |F_x(i,j,k)| > |F_x(i+1,j,k)| > |F_x(i+2,j,k)|$$

en el caso en que $|F_x(i,j,k)|\,h_x > |F_y(i,j,k)|\,h_y, |F_z(i,j,k)|\,h_z$, o bien que

$$|F_z(i,j,k-2)| < |F_z(i,j,k-1)| < |F_z(i,j,k)| > |F_z(i,j,k+1)| > |F_z(i,j,k+2)|$$

en el caso en que $|F_z(i,j,k)|\,h_z > |F_x(i,j,k)|\,h_x, |F_y(i,j,k)|\,h_y$.

Una vez tenemos los voxels identificados, aplicaremos alrededor de cada uno el lema 7.7 para obtener los parámetros del contorno (posición, vector normal y salto de intensidad). Esto significa que, para cada voxel borde, el método usará los valores de un entorno centrado en dicho pixel, cuya orientación variará en función de qué parcial es mayor.

Hay tres condiciones que deben cumplirse dentro de dicho entorno, o al menos acercarse lo más posible, para que la situación alrededor de esos voxels sea lo más parecido a la situación ideal sobre la que se ha desarrollado el método:

1) Dentro de ese entorno, el contorno puede ser aproximado como un plano.

2) Dentro de ese entorno, no debe aparecer ningún otro contorno diferente.

3) Dentro de ese entorno, la intensidad a ambos lados del contorno debe ser lo más homogénea posible.

Lógicamente, es difícil que las tres condiciones se cumplan exhaustivamente en todos los voxels borde, pero también es bastante probable que se cumplan en un grado alto. La primera condición fallará en zonas donde la curvatura del contorno sea muy alta. La segunda fallará en zonas donde haya dos bordes muy cercanos entre sí. La tercera fallará en los bordes de zonas con textura muy fina. En los demás casos, las condiciones se cumplirán.

Hay otra condición que también asumimos, y es que la imagen no tenga ruido, o que tenga muy poco. En capítulos posteriores abordaremos este problema, pero por ahora trabajaremos con imágenes sintéticas.

El pseudo-código de nuestro algoritmo podría ser el siguiente:

Funcion Contornos3D (delta)

```
Calcular Fx, Fy, Fz
Para todos los voxels (i,j,k) de la imagen
   Si Fx(i,j,k)*Fx(i,j,k)+Fy(i,j,k)*Fy(i,j,k)+Fz(i,j,k)*Fz(i,j,k)<delta*delta Continuar
   Si (|Fx(i,j,k)|hx>|Fy(i,j,k)|hy) y (|Fx(i,j,k)|hx>|Fz(i,j,k)|hz) Entonces
      Si |Fx(i,j,k)| no es maximo en su fila Continuar
      Aplicar lema 7.7 con la variable x como direccion principal
   Fin Si
   Si (|Fy(i,j,k)|hy>|Fx(i,j,k)|hx) y (|Fy(i,j,k)|hy>|Fz(i,j,k)|hz) Entonces
      Si |Fy(i,j,k)| no es maximo en su columna Continuar
      Aplicar lema 7.7 con la variable y como direccion principal
   Fin Si
   Si (|Fz(i,j,k)|hz>|Fx(i,j,k)|hx) y (|Fz(i,j,k)|hz>|Fy(i,j,k)|hy) Entonces
      Si |Fz(i,j,k)| no es maximo en su fila Continuar
      Aplicar lema 7.7 con la variable z como direccion principal
   Fin Si
Fin Para
```

7.10 Ejemplos

7.10.1 Contornos de primer orden

Al aplicar este algoritmo a una imagen ideal con un contorno de primer orden $F_{i,j,k}(A, B, a, b, c, e)$ (ecuación 7.5), como la que se mostraba en la figura 7.6, el resultado es exacto, independientemente de los valores que tengan los coeficientes a, b, c, e, A y B. Por el contrario, la técnica tradicional mediante convolución con máscaras H_x, H_y y H_z produce errores de diferentes magnitudes en función del valor de los coeficientes. En la tabla 7.1 se muestran los errores para contornos de distinta orientación , aplicando ambos métodos. Para el método tradicional se han tomado los valores de alfa y beta propuestos por L. Álvarez (expresión 7.3).

Todos los contornos tienen un salto de intensidad (módulo del vector normal) de 255 unidades. El error del módulo viene especificado en unidades de intensidad, y el de la dirección del vector normal en grados. La primera fila indica que, sea cual sea la orientación del contorno, el método propuesto no comete error alguno. Tampoco lo hace el método tradicional para planos paralelos a las direcciones principales, pero para cualquier otra orientación, los errores llegan a alcanzar los 4 grados en la dirección del vector normal, y las 24 unidades de diferencia en el salto de intensidad.

		Error del módulo				Error de la dirección			
Método	**Normal**	Media	Desv.	Mín.	Max.	Media	Desv.	Mín.	Max.
Propuesto	$(-,-,-)$	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Tradicional	$(1,0,0)$	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Tradicional	$(1,1,1)$	4.35	0.00	4.35	4.35	0.00	0.00	0.00	0.00
Tradicional	$(-4,3,6)$	10.5	9.27	0.74	24.0	3.24	0.64	2.32	3.99
Tradicional	$(0,1,4)$	9.32	6.87	2.93	17.6	2.67	0.33	2.33	3.08
Tradicional	$(3,10,7)$	7.73	7.22	1.35	24.1	3.42	0.64	2.13	4.18

Tabla 7.1: Errores cometidos por cada método al obtener el contorno de un plano

Para poder visualizar gráficamente estos resultados, se ha generado para cada voxel etiquetado como borde, un trozo de plano inscrito en él, cuya posición y orientación vienen dados por los parámetros detectados para el contorno en dicho voxel, y cuyo color representa el error cometido en la estimación de dicho plano (ver figura 7.20). De esta forma podemos apreciar mejor cómo el error es nulo con nuestro método en todos los voxels borde, mientras que con el método tradicional el error va variando a lo largo del contorno. Se ha utilizado el plano indicado en la última fila de la tabla 7.1. No se muestra la gráfica para el error de la posición con el método tradicional, ya que esta técnica no lo estima. De esta manera, para poder mostrar los errores en el salto de intensidad y en la dirección con esta técnica se han tomado "prestados" los valores de posición obtenidos por nuestro método.

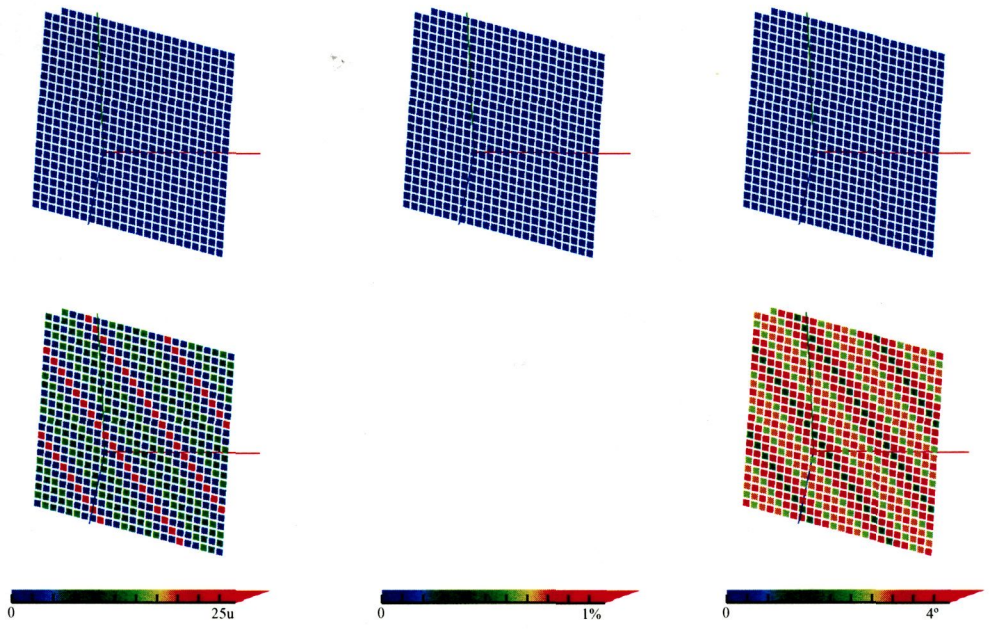


Figura 7.20: Error cometido al detectar el contorno de un plano con el vector $(3,10,7)$ como vector normal, utilizando el método propuesto (fila superior) y el método tradicional (fila inferior). De izquierda a derecha, error cometido en la estimación del salto de intensidad, posición y orientación del contorno.

	Error del módulo				Error de la dirección				Error de la posición			
Método	Media	Desv.	Mín.	Max.	Media	Desv.	Mín.	Max.	Media	Desv.	Mín.	Max.
Propuesto	0.00	0.00	0.00	0.00	0.15	0.08	0.00	0.53	0.65	0.32	0.01	1.50
Tradicional	6.98	6.00	0.00	21.8	2.98	1.08	0.00	4.85	–	–	–	–

Tabla 7.2: Errores cometidos por cada método al obtener el contorno de una esfera de radio 20

7.10.2 Contornos de segundo orden

Generemos ahora una imagen con un contorno de segundo orden, concretamente una esfera maciza de intensidad clara en el interior y oscura en el exterior, de radio 20 voxels. En la tabla 7.2 podemos apreciar los errores cometidos por el método propuesto y por el tradicional. El método tradicional sigue teniendo errores grandes tanto en el módulo (error medio de 7 unidades en cada voxel) como en la orientación del vector normal (casi 3 grados de error medio en el cálculo de la dirección normal). El método propuesto ya no es exacto como antes, pero los errores cometidos son bastante pequeños (0.15 grados en la dirección y 0.65% en la posición sub-voxel).

En la figura 7.21 podemos ver con detalle el error cometido en cada voxel. Puede apreciarse cómo el método tradicional logra su mejor resultado a lo largo de las 3 direcciones principales (zonas azules en los extremos superior, izquierdo y derecho de las esferas), pero comete un error mayor cuanto más se aleja de ellas. El método propuesto, por el contrario, es igual de efectivo en cualquier orientación.

7.11 Conclusiones

Se ha desarrollado un esquema capaz de encontrar en imágenes ideales sintéticas la orientación y localización sub-voxel de un contorno plano de forma exacta, así como el salto de intensidad producido a ambos lados del borde. Cuando el contorno no sea plano, los errores más importantes ocurrirán en el cálculo de la posición sub-voxel del contorno (como puede apreciarse en la esfera de la columna central de la figura 7.21), debido a la curvatura presente en el contorno de la imagen, que nuestro método aún es incapaz de detectar. Por ello, en el próximo capítulo propondremos un nuevo método que pueda detectar también la curvatura del contorno, para poder detectar la superficie de éste con mayor precisión.

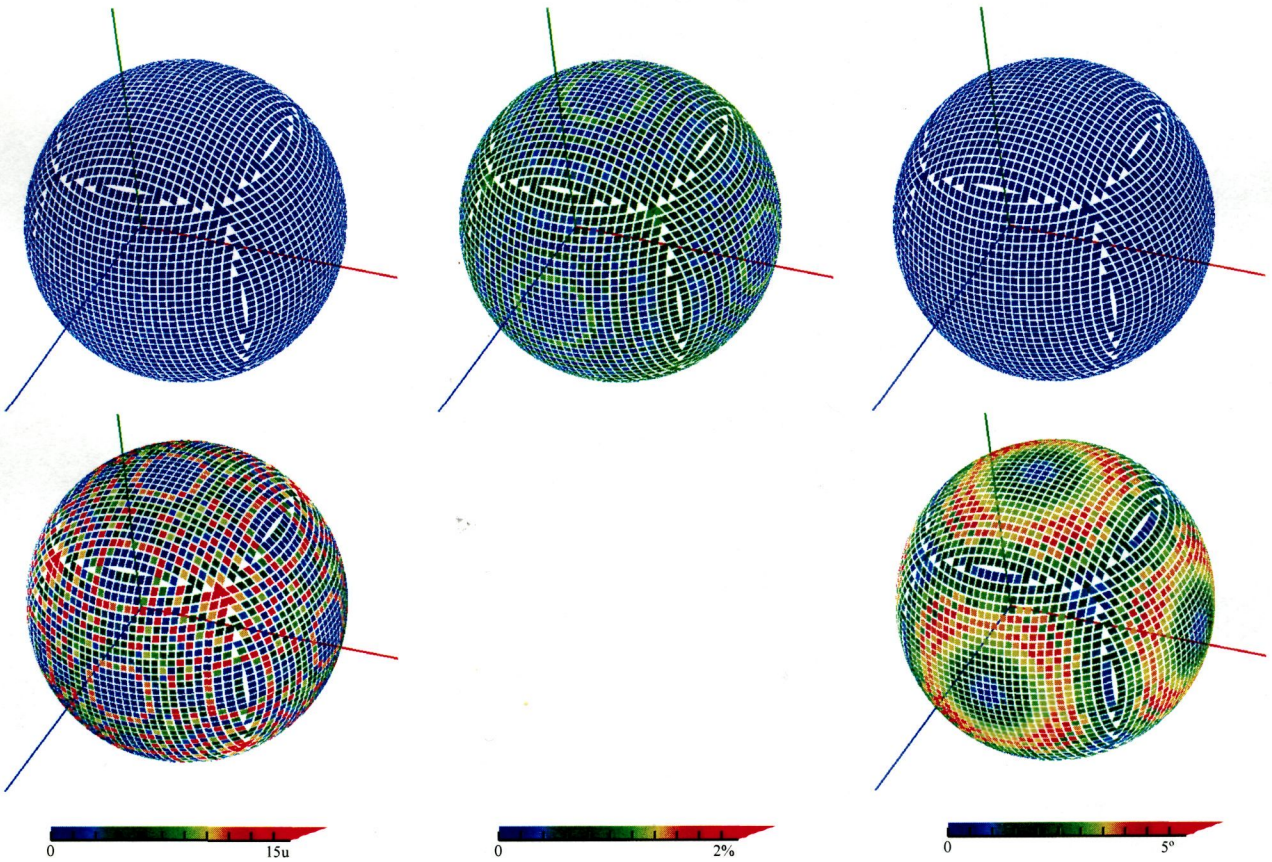


Figura 7.21: Error cometido al detectar el contorno de una esfera de radio 20 utilizando el método propuesto (fila superior) y el método tradicional (fila inferior). De izquierda a derecha, error cometido en la estimación del salto de intensidad, posición y orientación del contorno.

Capítulo 8

Contornos de segundo orden

- Ahora habrá que pasar a grado 2.
- Bienvenido Mr. Paraboloide....

MAYO 2003

Acabamos de ver en el capítulo anterior un método que calcula la posición y orientación del contorno que pasa por un voxel de forma exacta (al menos en imágenes ideales en el sentido de nuestras hipótesis de partida). Pero existe otro parámetro característico del contorno que resulta necesario estimar en muchas aplicaciones: la curvatura. A diferencia del caso bidimensional, en una imagen 3D el contorno viene dado por una superficie en lugar de una curva, y por tanto en cada punto no hay un único valor de curvatura sino infinitas, dependiendo de la dirección en la que miremos.

Si observamos la figura 8.1a, vemos cómo para cualquier dirección que prefijemos sobre el plano tangente a la superficie en dicho punto, tendremos una curvatura diferente. En realidad, la curvatura de la superficie en una dirección dada (perteneciente siempre al plano tangente) viene dada por la curvatura de la curva intersección entre la superficie y el plano normal al plano tangente que contiene el vector que representa dicha dirección. De todos los valores de curvatura posibles existentes en un punto, se denominan **curvaturas principales** a los valores máximo y mínimo de todos ellos. Además, es un resultado clásico de la geometría el hecho de que las direcciones en las que se producen ambos valores son siempre perpendiculares entre sí para cualquier superficie [ROG89].

Así por ejemplo, sobre la superficie de una esfera hay un único valor de curvatura, con lo cual las curvaturas principales coincidirían, y la dirección de sus vectores sería irrelevante. Para un toroide, la máxima curvatura ($K_{\max}$) será siempre tangente al perímetro de la sección, y la mínima ($K_{\min}$) apuntará en la dirección del toroide, como se ve en la figura 8.1b.

Por lo tanto, el objetivo sería desarrollar un método que sea capaz de estimar con precisión para cada voxel borde los tres vectores siguientes:

a) el vector gradiente, cuya dirección es normal a la superficie en ese punto, y cuyo módulo representa el salto de intensidad entre los voxels a ambos lados del contorno,

b) el vector de máxima curvatura, cuya dirección apunta hacia donde la superficie tiene una curvatura mayor, y cuyo módulo representa la magnitud de dicha curvatura,

c) el vector de mínima curvatura, cuya dirección apunta hacia donde la superficie tiene una curvatura menor, y cuyo módulo representa la magnitud de dicha curvatura.

Estos tres vectores deberán ser ortogonales entre sí. Además de dichos vectores, el método deberá estimar también la posición exacta en el interior del voxel del punto del contorno donde los tres vectores han sido

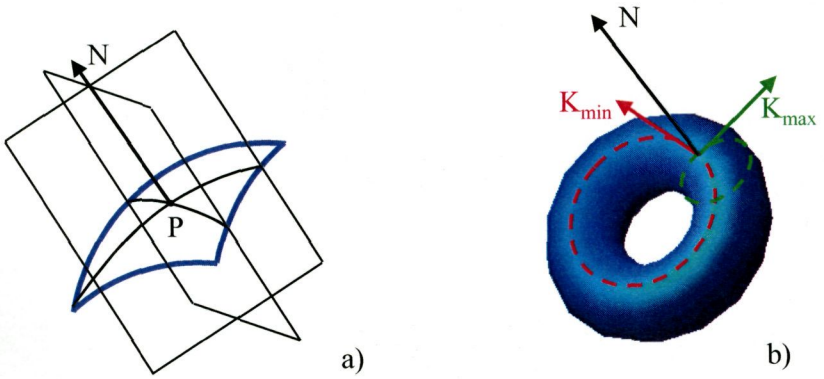


Figura 8.1: a) En un punto de una superficie existen tantas curvaturas como planos que contengan al vector normal a la superficie en ese punto. b) direcciones normal (en negro), de mínima curvatura (en rojo) y de máxima curvatura (en verde) sobre un punto de la superficie de un toroide

estimados.

El interés de estos vectores radica en la enorme cantidad de información que se aporta sobre la geometría del objeto. Por ejemplo, en imágenes angiográficas tridimensionales, donde los objetos principales presentes son vasos sanguíneos (ver figura 8.2a y b), poder conocer con precisión en cada voxel borde los tres vectores anteriores (figura 8.2d) nos permitiría saber, entre otras cosas, en qué dirección está orientado el vaso (dirección de curvatura mínima) y cual es su diámetro aproximado en ese punto (a partir de la curvatura máxima). Un estudio en profundidad sobre la detección de vasos en imágenes 3D puede verse en la tesis de Krissian [KRI00a].

Esta información puede usarse como ayuda al diagnóstico médico, como por ejemplo, la detección de estenosis (disminución del diámetro en una zona del vaso) [KRI00b, YIM00], o bien para hacer una reconstrucción sintética de la geometría de la vasculatura [KRI00c, BUH04], como la mostrada en la figura 8.2f, que permita una visualización más realista y precisa que la obtenida mostrando simplemente los voxels etiquetados como borde[1]. Una extensa revisión de las técnicas y algoritmos usados en la detección de vasos así como su utilidad médica, tanto en imágenes 2D como 3D, puede verse en [KIR03].

En este capítulo analizaremos primeramente las técnicas convencionales para estimar estos vectores, y posteriormente propondremos un nuevo método que estime dichos valores con la mayor exactitud posible. Finalmente lo aplicaremos a varias imágenes sintéticas.

8.1 Cálculo tradicional de curvaturas en imágenes 3D

Existen dos formas principales de abordar el problema. Una consiste en realizar la estimación a partir de las segundas derivadas de la imagen. La otra exige una representación analítica previa de la función que representa la imagen. Analicemos ambas técnicas por separado.

[1] La presente tesis está centrada en la etapa de estimación (figura 8.2d). La reconstrucción se ha mostrado en las figuras 8.2e y f sólamente a modo explicativo.

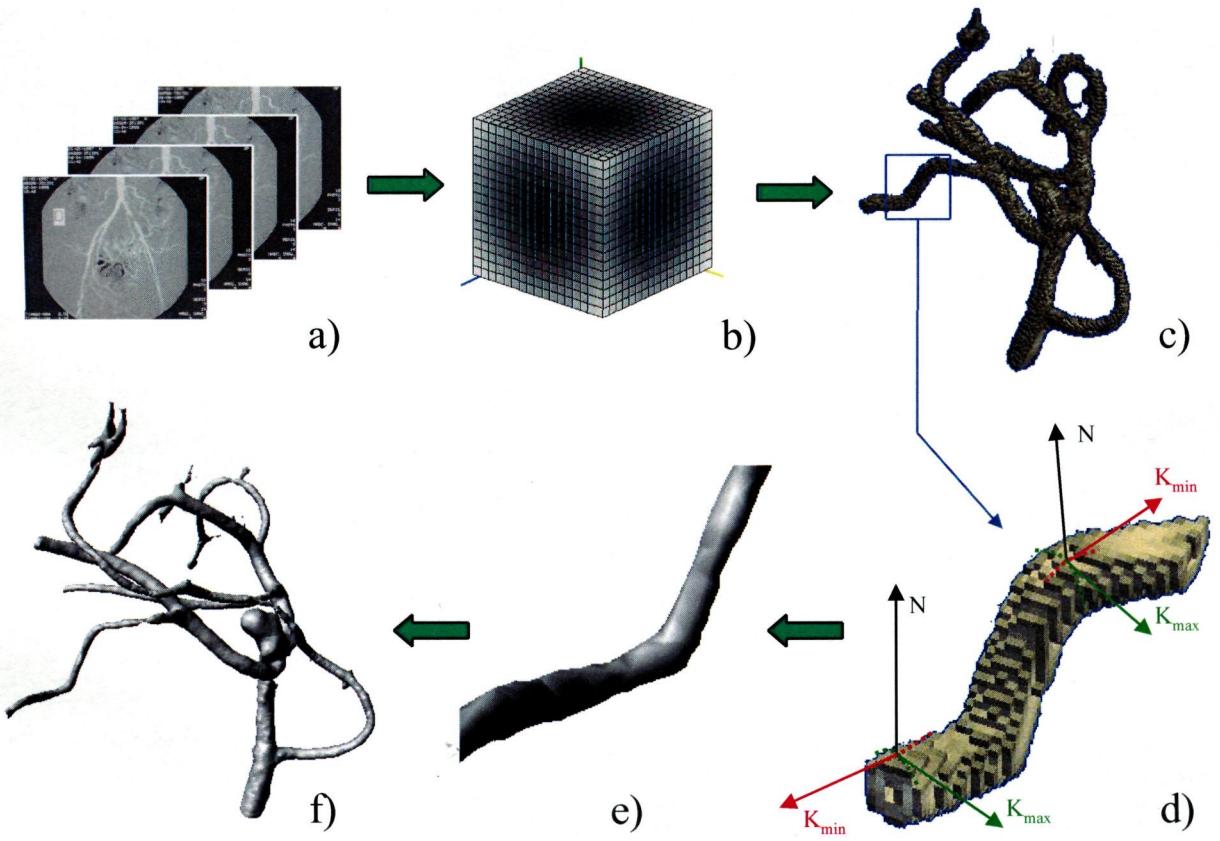


Figura 8.2: a,b) angiografía tridimensional; c) voxels borde; d) detalle de un vaso, con la estimación del vector normal y las curvaturas; e) reconstrucción del vaso, a partir de los vectores obtenidos; f) reconstrucción de la vasculatura completa de la imagen

8.1.1 A partir de las segundas derivadas de la imagen

Al igual que en el caso 2D, las curvaturas están relacionadas con las segundas derivadas. Por lo tanto, el método usual es calcular primero estas derivadas en cada voxel borde a partir de máscaras de convolución [MON95, THI95, ALV01]. Una vez calculadas las segundas derivadas, obtenemos la matriz hessiana para ese voxel, la cual viene dada por la expresión

$$H(x,y,z) = \begin{pmatrix} f_{xx} & f_{xy} & f_{xz} \\ f_{yx} & f_{yy} & f_{yz} \\ f_{zx} & f_{zy} & f_{zz} \end{pmatrix}$$

donde $f(x,y,z)$ es la imagen 3D.

A partir de la matriz hessiana, existen varias formas de obtener las direcciones y valores de las curvaturas principales.

Técnica propuesta por Monga en [MON95] Recordemos que al considerar la imagen como una función de tres variables, las curvaturas que estamos estimando son las de la superficie de nivel que pasa por el punto. De esta forma, se calcula primero la curvatura de la isosuperficie en una cierta dirección u (perteneciente al plano tangente a la isosuperficie) con la expresión

$$K(u) = -\frac{u^T H u}{\|\nabla F\|}$$

donde H es la matriz hessiana y ∇F es el vector gradiente, ambos evaluados en ese voxel. A continuación, se buscan los extremos de la función $K(u)$ para obtener las curvaturas principales, derivando en función del ángulo θ alrededor del vector normal. Así, se obtienen los ángulos

$$\begin{aligned} \theta_1 &= \frac{1}{2}\arctan\frac{2a^T H b}{a^T H a - b^T H b} \\ \theta_2 &= \theta_1 + \frac{\pi}{2} \end{aligned}$$

siendo a y b dos vectores ortonormales del plano tangente. A partir de estos ángulos se obtienen las direcciones de las curvaturas principales, con las siguientes expresiones

$$\begin{aligned} u_1 &= \cos\theta_1 a + \sin\theta_1 b \\ u_2 &= \cos\theta_2 a + \sin\theta_2 b \end{aligned}$$

Técnica propuesta por Álvarez en [ALV01] La segunda forma consiste en calcular primero el desarrollo de Taylor de la imagen $F(X)$ (siendo X el vector de coordenadas del voxel), pero restringido al plano tangente a la isosuperficie que pasa por dicho voxel, cuya expresión resulta ser

$$f(X + Ph) = f(X) + h^T P^T H P h$$

donde H vuelve a ser el hessiano evaluado en el voxel, y donde Ph representa la proyección de un cierto vector h sobre el plano tangente a la isosuperficie, cuya expresión es

$$Ph = \left(Id - \frac{\nabla f (\nabla f)^T}{\|\nabla f\|^2}\right) h$$

A continuación, y considerando la matriz $A = P^T H P$, se demuestra que A posee un autovalor igual a 0, cuyo autovector asociado viene dado por la dirección del vector gradiente, y que los otros dos autovectores se corresponden con las direcciones de las curvaturas principales. Dichas curvaturas vienen dadas por las expresiones

$$K_i = \frac{\lambda_i}{\|\nabla f\|} \quad i = 1, 2$$

donde λ_1 y λ_2 son los autovalores no nulos de A.

8.1.2 A partir de la expresión de una función que aproxime a la imagen

Otro método diferente, propuesto por Brejl en [BRE00], para estimar tanto el gradiente como las curvaturas consiste en hacerlo analíticamente a partir de la expresión de una cierta función $p(x, y, z)$, cuya discretización

$P(x_i, y_j, z_k)$ es la que mejor aproxima las intensidades de los voxels en una cierta vecindad alrededor del voxel seleccionado. Es decir, de nuevo vuelve a suponerse que la imagen de entrada (anterior a la discretización), $f(x, y, z)$, es continua y derivable en todo su dominio, y lo que se busca es encontrar la función $p(x, y, z)$ dada por

$$\begin{aligned} p(x,y,z) &= k_0 + k_1 x + k_2 y + + k_3 z + k_4 x^2 + k_5 xy + k_6 xz + k_7 yz + k_8 y^2 + k_9 z^2 + \\ &\quad + k_{10} x^3 + k_{11} x^2 y + k_{12} x^2 z + k_{13} xy^2 + k_{14} xz^2 + k_{15} y^2 z + k_{16} yz^2 + k_{17} y^3 + k_{18} z^3 + k_{19} xyz \end{aligned}$$

que mejor la aproxime en un cierto entorno, considerando un sistema de referencia centrado en el centro geométrico del voxel que estamos estudiando.

Para ello, y haciendo uso del mismo modelo de adquisición expuesto en el capítulo anterior (ecuación 7.4), se considera que la función discretizada adquirida en la imagen viene dada por la expresión

$$P(x_i, y_j, z_k) = \frac{1}{h_x h_y h_z} \int_{z_k - h_z/2}^{z_k + h_z/2} \int_{y_j - h_y/2}^{y_j + h_y/2} \int_{x_i - h_x/2}^{x_i + h_x/2} p(x, y, z) dx dy dz$$

donde h_x, h_y y h_z indican la longitud de un voxel en cada dirección. Resolviendo la integral de forma analítica obtenemos una expresión general para $P(x_i, y_j, z_k)$. A continuación, se minimiza el error cuadrático medio cometido al aproximar esta función en una cierta vecindad del voxel central, el cual viene dado por la expresión

$$e(k_0, k_1, ..., k_{19}) = \sum_i \sum_j \sum_k \left(P(x_i, y_j, z_k) - F(i, j, k)\right)^2$$

En este caso, la sumatoria indica la extensión de la vecindad considerada. Como resultado de la minimización, se obtienen los valores óptimos para los coeficientes k_i de la función $p(x, y, z)$. Finalmente, se calculan las expresiones de los parámetros del contorno (gradiente y curvaturas) a partir de la expresión analítica de la función $p(x, y, z)$, y se evalúan en el centro geométrico del voxel en cuestión, es decir, en el punto $(0, 0, 0)$.

8.1.3 Errores de precisión

La exactitud del primer método (a partir de las segundas derivadas) está ligada a la precisión obtenida en la convolución con las máscaras en el cálculo de las primeras y segundas derivadas de la imagen. Y ya demostramos en el capítulo anterior que la estimación no es exacta en todos los casos. Por otro lado, hay que tener en cuenta que las curvaturas son valores extremadamente sensibles, con lo cual variaciones mínimas de la intensidad en los voxels producen grandes variaciones en los valores de las curvaturas. Es por ello que en la imagen se suele realizar algún tipo de suavizado previo antes de realizar la convolución, con lo cual los valores de las derivadas pierden precisión.

Los resultados obtenidos por el segundo método suelen ser bastante satisfactorios, sobre todo en zonas donde la hipótesis de que la función $f(x, y, z)$ es continua y derivable parece cumplirse. Éste es el caso de zonas con intensidad homogénea, alejadas de las superficies presentes en la imagen, o zonas con contornos ligeramente difuminados. Sin embargo, no determinan la localización sub-voxel del contorno, y tampoco son exactos en imágenes ideales en el sentido de nuestras hipótesis de partida ($f(x, y, z)$ es discontinua en los bordes). Además, en el aspecto computacional, la minimización de la función de error, teniendo en cuenta que es una función de 20 coeficientes, es bastante costosa.

Por otro lado, ambos métodos evalúan los parámetros del borde en el punto $(0, 0, 0)$, y eso significa que interpretan el contorno como la superficie de nivel de $p(x, y, z)$ que pasa por el centro del voxel. Pero superficies de nivel que atraviesen el voxel seleccionado hay infinitas, y cada una de ellas tiene infinitos puntos que la forman dentro del voxel. Y cada uno de esos puntos tendrá valores de gradiente y curvaturas diferentes. Es cierto que serán valores bastante similares, pero no son exactos en un sentido riguroso.

8.2 Detector de segundo orden para las superficies $y = f(x, z)$

Vamos a continuación a desarrollar un método que sí nos permita detectar con exactitud las curvaturas principales, el vector normal y la posición interior al voxel de las superficies presentes en la imagen, al menos en imágenes ideales en el sentido de nuestras hipótesis iniciales. Dicho método será una extrapolación del que desarrollamos para detectar bordes de segundo orden en imágenes 2D (ver sección 3.3).

En el caso bidimensional, partíamos de una aproximación local de segundo grado, similar a la usada en el polinomio de Taylor, para la línea del contorno interior al pixel (ecuación 3.5). Aquí haremos lo mismo, pero usando la expresión de Taylor para una función de dos variables. Por lo tanto, sea $y = f(x, z)$ una función de dos variables[2], y queremos encontrar su aproximación alrededor del origen de coordenadas, como se ve en la figura 8.3.

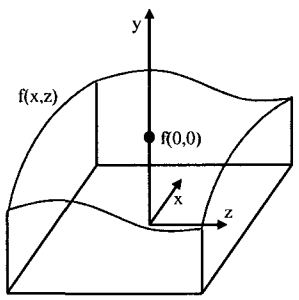


Figura 8.3: Dentro del voxel, el contorno será aproximado por una superficie de segundo grado

El polinomio de Taylor de grado 2 centrado en $(0, 0)$ es el siguiente:

$$\begin{aligned} P_2(x,z) &= f(0,0) + f_x(0,0)x + f_z(0,0)z + \frac{1}{2}\left(f_{xx}(0,0)x^2 + 2f_{xz}(0,0)xz + f_{zz}(0,0)z^2\right) = \\ &= a + bx + cz + dx^2 + fxz + gz^2 \end{aligned}$$

Es decir, es una ecuación de grado 2 en dos variables, lo que representa en realidad un **paraboloide**. Por lo tanto, el método que proponemos consiste en suponer que, en la vecindad del voxel, el borde se asemeja a un paraboloide. Así, en primer lugar trataremos de encontrar los 6 coeficientes de la expresión que mejor se ajustan a los valores de la imagen, y una vez obtenidos, calcularemos todos sus parámetros (posición, vector normal y curvaturas principales) de forma analítica a partir de dicha expresión.

8.2.1 Definición del entorno del voxel

Primero hay que definir claramente el sentido y la orientación de las variables, como se ve en la figura 8.4. El origen de coordenadas lo colocaremos en el centro geométrico del voxel que queremos estudiar (el voxel central en la figura), el cual tendrá por coordenadas $(0, 0, 0)$. El valor de y crece hacia arriba. El valor de z crece hacia la derecha. El valor de x crece hacia atrás.

De esta forma, si llamamos F a la imagen 3D de voxels, estaremos diciendo que para el caso de la figura se cumple que $|F_y| > |F_x|, |F_z|$, y por lo tanto, para este caso, la superficie puede considerarse como una función explícita, con la y despejada en función de las variables x y z. Lógicamente, una vez resuelto este caso, todos los demás se resolverán de forma análoga intercambiando las variables.

[2] Inicialmente sólo consideraremos superficies donde las pendientes y/x e y/z están entre 0 y 1, y por ello puede despejarse la variable y en función de las variables x y z.

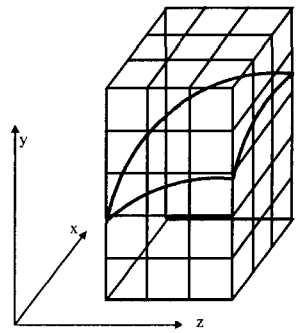


Figura 8.4: El sistema de coordenadas tiene la orientación que se indica, pero está situado en el centro geométrico del voxel central

En cuanto al tamaño de lo que estamos considerando "vecindad local", en la figura 8.4 hemos usado un entorno o subimagen de tamaño 3x5x3, de forma similar al caso bidimensional, donde se usaba una ventana 3x5. Recordemos que estas dimensiones se debían a la imposición de que, al menos en el caso lineal para pendientes entre 0 y 1, la línea del contorno cruzara siempre el entorno de izquierda a derecha, sin tocar nunca los límites inferior ni superior de la subimagen.

Sin embargo, al trabajar en 3D, y aún considerando el caso lineal, no podemos garantizar que la superficie del contorno no toque el suelo o el techo de dicho entorno. Es decir, considerando que la superficie fuese un plano (aproximación lineal), y que en el caso peor las pendientes de x y de z fuesen cercanas a 1 (45^o), vemos que en la figura 8.5a (5 voxels de altura) no podemos garantizar que el plano quede completamente dentro de la subimagen, por lo que necesitaríamos un entorno con al menos 7 voxels de altura como se ve en la figura 8.5b para que cupiera totalmente.

Aun así, si consideramos una aproximación cuadrática, podría resultar que tampoco cupiera, llegando a un caso extremo como el que estábamos hablando (pendientes en x y en z cercanas a 1), con una cierta curvatura, en donde el paraboloide se saliese del entorno, como se ve en la figura 8.5c.

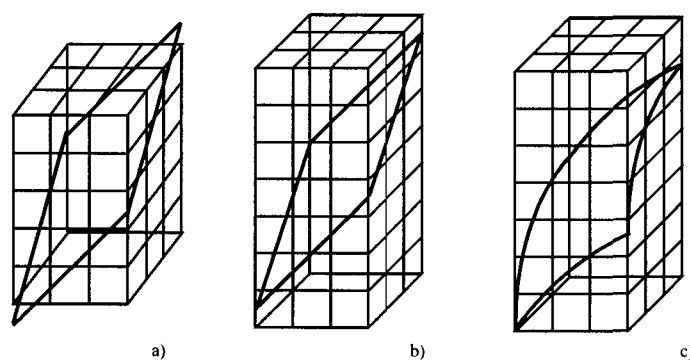


Figura 8.5: Un entorno 3x5x3 no garantiza que el plano corte el suelo (a), pero un entorno 3x7x3 sí lo garantiza (b), aunque no para un paraboloide (c).

Recordemos que la importancia de que la superficie no toque el suelo de la subimagen es para que, después, cuando integremos cada una de las columnas, resulten expresiones lineales, puesto que de lo contrario el sistema a resolver para hallar los coeficientes del paraboloide sería bastante complejo. Sin embargo, aumentar la altura de la subimagen a 9 voxels significaría estar usando voxels muy alejados para deducir los parámetros del voxel central, y en realidad esos casos extremos no se darán con mucha frecuencia. Así que, en lo sucesivo, trabajaremos con subimágenes de 3x7x3.

8.2.2 Cálculo de los coeficientes del paraboloide

Ahora que ya tenemos definido con precisión nuestro entorno, podemos plantear el sistema de ecuaciones que nos llevará a obtener los valores para el paraboloide, cuya expresión recordemos que es

$$y = a + bx + cz + dx^2 + fxz + gz^2$$

Planteamiento del sistema

Son 6 los coeficientes que han de calcularse, y 9 las columnas de las que disponemos en el plano xz sobre las cuales calcular. Una opción podría ser plantear las 9 ecuaciones, y buscar los valores de los 6 coeficientes que minimizan el error con respecto a las 9 ecuaciones. Sin embargo, nosotros plantearemos una opción diferente, consistente en ignorar algunas de las columnas, concretamente las pertenecientes a los voxels $(-1, 0, -1)$ y $(1, 0, 1)$, tal y como se aprecia en la figura 8.6.

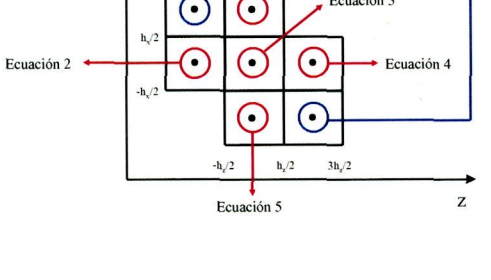


Figura 8.6: Para obtener los coeficientes del paraboloide se va a resolver el sistema de 6 ecuaciones planteado, donde la ecuación 6 es una suma de los valores de las dos columnas de color azul.

¿Por qué quitar esas columnas y no otras? Realmente nos interesa ignorar aquéllas donde la superficie alcance valores extremos, para minimizar la probabilidad de que la superficie pueda tocar el suelo o el techo de la subimagen en dichas columnas, y esto sucederá en las columnas $(-1, .., -1)$ y $(1, .., 1)$ cuando las parciales F_x y F_z tengan el mismo signo, y en las columnas $(-1, .., 1)$ y $(1, .., -1)$ cuanto el signo sea distinto.

La razón principal de usar esta alternativa es que, tal y como explicábamos en la sección anterior, existe un problema cuando la superficie alcanza el suelo de la subimagen (o el techo), y en esos casos, esto sucederá precisamente en las columnas que se han quitado. Con lo cual esto permite poder detectar con el método superficies con curvatura un poco mayores que antes. Nótese que no perdemos información, ya que en realidad para formar las 6 ecuaciones bastaría con sólo el mismo número de columnas, y aún seguimos teniendo 7 de las 9 que había.

Planteamiento de las ecuaciones

Recordemos del capítulo anterior que la intensidad de un voxel (i, j, k) por el cual atraviesa una superficie viene dada por la expresión

$$F(i,j,k) = B + \frac{A-B}{h_x h_y h_z} V_{i,j,k}$$

donde $V_{i,j,k}$ indica el volumen interior al voxel (i, j, k) que cae bajo dicha superficie, y A y B son las intensidades por debajo y por encima de la superficie respectivamente. Por otro lado, la suma de los valores de los voxels de una misma columna, siempre y cuando se cumpla que la superficie no toque los planos inferior y superior de dicha columna, viene dada por la expresión

$$S = Bm + \frac{A-B}{h_x h_y h_z} V$$

donde m es el número de voxels de la columna, y V es el volumen interior a la columna que se encuentra bajo la superficie.

Por lo tanto, llamemos $S_1, ..., S_6$ a las sumas de los valores de las columnas siguientes:

$$\begin{array}{lll} S_1 = \sum_{j=-3}^{3} F_{1,j,0} & S_2 = \sum_{j=-3}^{3} F_{0,j,-1} & S_3 = \sum_{j=-3}^{3} F_{0,j,0} \\ S_4 = \sum_{j=-3}^{3} F_{0,j,1} & S_5 = \sum_{j=-3}^{3} F_{-1,j,0} & S_6 = \sum_{j=-3}^{3} (F_{1,j,-1} + F_{-1,j,1}) \end{array}$$

donde estamos considerando el caso $F_x F_z > 0$.

Tomando estas 6 sumas como constantes, el sistema de 6 ecuaciones planteado es el siguiente:

$$\begin{cases} S_n = 7B + \dfrac{A-B}{h_x h_y h_z} V_n & \forall n = 1, .., 5 \\ S_6 = 14B + \dfrac{A-B}{h_x h_y h_z} V_6 \end{cases}$$

donde

$$\begin{aligned}
V_1 &= \int_{h_x/2}^{3h_x/2} \int_{-h_z/2}^{h_z/2} P(x,z)dzdx = ah_x h_z + \frac{7}{2} h_x h_y h_z + bh_x^2 h_z + \frac{13}{12} dh_x^3 h_z + \frac{1}{12} gh_x h_z^3 \\
V_2 &= \int_{-h_x/2}^{h_x/2} \int_{-3h_z/2}^{-h_z/2} P(x,z)dzdx = ah_x h_z + \frac{7}{2} h_x h_y h_z - ch_x h_z^2 + \frac{1}{12} dh_x^3 h_z + \frac{13}{12} gh_x h_z^3 \\
V_3 &= \int_{-h_x/2}^{h_x/2} \int_{-h_z/2}^{h_z/2} P(x,z)dzdx = ah_x h_z + \frac{7}{2} h_x h_y h_z + \frac{1}{12} dh_x^3 h_z + \frac{1}{12} gh_x h_z^3 \\
V_4 &= \int_{-h_x/2}^{h_x/2} \int_{h_z/2}^{3h_z/2} P(x,z)dzdx = ah_x h_z + \frac{7}{2} h_x h_y h_z + ch_x h_z^2 + \frac{1}{12} dh_x^3 h_z + \frac{13}{12} gh_x h_z^3 \\
V_5 &= \int_{-3h_x/2}^{-h_x/2} \int_{-h_z/2}^{h_z/2} P(x,z)dzdx = ah_x h_z + \frac{7}{2} h_x h_y h_z - bh_x^2 h_z + \frac{13}{12} dh_x^3 h_z + \frac{1}{12} gh_x h_z^3 \\
V_6 &= \int_{h_x/2}^{3h_x/2} \int_{-3h_z/2}^{-h_z/2} P(x,z)dzdx + \int_{-3h_x/2}^{-h_x/2} \int_{h_z/2}^{3h_z/2} P(x,z)dzdx = \\
&= 2ah_x h_z + 7h_x h_y h_z + \frac{13}{6} dh_x^3 h_z + \frac{13}{6} gh_x h_z^3 - 2fh_x^2 h_z^2
\end{aligned}$$

y donde hemos llamado

$$P(x,z) = a + bx + cz + dx^2 + fxz + gz^2 + 7h_y/2$$

Se trata de un sistema lineal de fácil resolución, cuyas soluciones son:

$$\begin{aligned}
a &= \tfrac{h_y}{24(A-B)}\,(28S_3 - S_1 - S_2 - S_4 - S_5 - 84A - 84B)\\
b &= \tfrac{h_y}{2h_x(A-B)}\,(S_1 - S_5)\\
c &= \tfrac{h_y}{2h_z(A-B)}\,(S_4 - S_2)\\
d &= \tfrac{h_y}{2(A-B)h_x^2}\,(S_1 + S_5 - 2S_3)\\
f &= \tfrac{h_y}{2(A-B)h_xh_z}\,(S_1 + S_2 + S_4 + S_5 - S_6 - 2S_3)\\
g &= \tfrac{h_y}{2(A-B)h_z^2}\,(S_2 + S_4 - 2S_3)
\end{aligned}$$

Como los valores $S_1, ..., S_6$ son conocidos, ya que se calculan a partir de los valores de los voxels, aún nos faltaría conocer otros valores, como son las dimensiones de un voxel, h_x, h_y, y h_z, y las intensidades a ambos lados de la superficie, A y B.

Normalmente el valor de los h suele ser irrelevante, por lo que se suele usar el caso $h_x = h_y = h_z = 1$, con lo cual las expresiones se simplifican. En otras ocasiones, h_z suele diferir de los otros dos. Téngase en cuenta que algunos dispositivos forman la imagen 3D como una secuencia de imágenes 2D en el plano xy, espaciadas a lo largo del eje z. Esto significa que los valores h_x y h_y indican las dimensiones de un pixel de cualquier imagen (generalmente cuadrados) y h_z indica la distancia en z que separa dos imágenes consecutivas. Dicha distancia es conocida por el personal que utiliza el dispositivo, e incluso en muchos de ellos es ajustable. En estos casos, lo más sencillo es hacer $h_x = h_y = 1$ y hacer h_z igual a la relación de distancias entre el espaciado de las imágenes y el tamaño de un pixel.

Estimación del salto de intensidad

Necesitamos también una estimación inicial de las intensidades A y B a ambos lados de la superficie. Al igual que en el caso 2D, lo más sencillo es utilizar los voxels que con toda seguridad sabemos que no van a tocar la superficie del contorno, y realizar un promedio. Atendiendo a la figura 8.5c, podemos tomar los voxels de las esquinas del entorno que más alejadas estén de la superficie para realizar dichas estimaciones, tal y como se muestra en la figura 8.7.

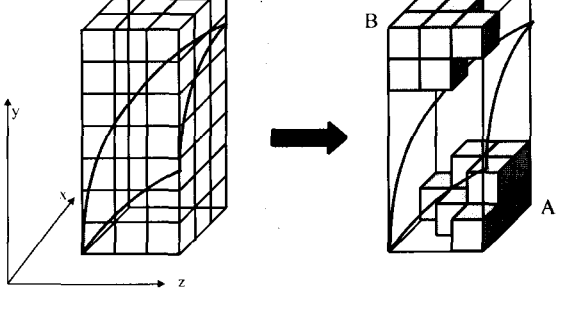


Figura 8.7: Para estimar las intensidades A y B usamos los 9 voxels de las esquinas más alejadas de la superficie

En realidad, las esquinas elegidas estarán en función del signo de las derivadas parciales. Por ejemplo, el caso mostrado en la figura se corresponderá con el caso $F_y < 0$ y $F_x, F_z > 0$ (suponiendo $A > B$), o bien con el caso $F_y > 0$ y $F_x, F_z < 0$ (suponiendo $A < B$). La expresión final para todos los subcasos posibles donde F_y es la derivada parcial de mayor absoluto es la siguiente:

$$\begin{aligned} A &\approx \frac{1}{9}\left(\begin{array}{c} F_{-\alpha,-3,\beta} + F_{-\alpha,-3,0} + F_{-\alpha,-3,-\beta} + F_{0,-3,0} + F_{0,-3,-\beta} + F_{\alpha,-3,-\beta} + \\ + F_{-\alpha,-2,0} + F_{-\alpha,-2,-\beta} + F_{0,-2,-\beta} + F_{-\alpha,-1,-\beta} \end{array} \right) \\ B &\approx \frac{1}{9}(F_{\alpha,3,-\beta} + F_{\alpha,3,0} + F_{\alpha,3,\beta} + F_{0,3,0} + F_{0,3,\beta} + F_{-\alpha,3,\beta} + F_{\alpha,2,0} + F_{\alpha,2,\beta} + F_{0,2,\beta} + F_{\alpha,1,\beta}) \end{aligned}$$

donde

$$\alpha = \begin{cases} 1 & \text{si } F_x F_y \geq 0 \\ -1 & \text{si } F_x F_y < 0 \end{cases} \qquad \beta = \begin{cases} 1 & \text{si } F_z F_y \geq 0 \\ -1 & \text{si } F_z F_y < 0 \end{cases}$$

Caso $F_x F_z < 0$

Cuando el producto $F_x F_z$ sea negativo, la expresión de S_6 cambiaría, ya que las columnas a considerar serían ahora las $(-1, .., -1)$ y $(1, .., 1)$, es decir

$$S_6 = \sum_{j=-3}^{3} (F_{1,j,1} + F_{-1,j,-1})$$

Por tanto, el volumen bajo la superficie en dichas columnas pasaría a ser

$$\begin{aligned} V_6 &= \int_{-3h_x/2}^{-h_x/2} \int_{-3h_z/2}^{-h_z/2} P(x,z)dzdx + \int_{h_x/2}^{3h_x/2} \int_{h_z/2}^{3h_z/2} P(x,z)dzdx = \\ &= 2ah_x h_z + 7h_x h_y h_z + \frac{13}{6} d h_x^3 h_z + \frac{13}{6} g h_x h_z^3 + 2f h_x^2 h_z^2 \end{aligned}$$

donde la única diferencia con la expresión anterior es el signo del último sumando. Al resolver el nuevo sistema, las expresiones para el paraboloide son idénticas, a excepción del coeficiente f que pasa a tener el signo invertido.

8.2.3 Cálculo de los parámetros del contorno

Acabamos de ver que, usando una subimagen de 3x7x3 centrada en el voxel, podemos obtener de forma sencilla la expresión para el paraboloide que mejor aproxima la superficie del contorno. Ahora usaremos dicha expresión para obtener los datos que queríamos, que son: posición, vector normal y curvaturas principales del contorno sobre la vertical central del voxel. El salto de intensidad a ambos lados del contorno es inmediato, ya que simplemente es $A - B$.

La posición y el vector normal también son fáciles de calcular. La posición viene dada por el coeficiente a del paraboloide. El vector normal sobre un punto del plano (x, z) viene dado por la expresión

$$N(x,z) = \begin{bmatrix} -b - 2dx - fz & 1 & -c - fx - 2gz \end{bmatrix}$$

Como estamos interesados en la normal sobre el punto $(0, 0)$, ésta es

$$N = \begin{bmatrix} -b & 1 & -c \end{bmatrix} \tag{8.1}$$

Sin embargo, el cálculo de las curvaturas en 3D es bastante más complicado que en el caso 2D, como ya se comentó al principio del capítulo. Para llevarlo a cabo vamos primero a utilizar un sistema de referencia diferente.

Cambio del sistema de referencia

Para poder calcular la curvatura del paraboloide en el punto $(0, a, 0)$, lo primero es cambiar a un nuevo sistema de coordenadas, $\{u, v, w\}$, donde el eje v coincida con el vector normal a la superficie en dicho punto, para poder luego encontrar la diferentes curvas de forma más fácil. Es decir, nuestro nuevo sistema será como se muestra en la figura 8.8. El origen del nuevo sistema lo pondremos sobre el punto mismo en cuestión.

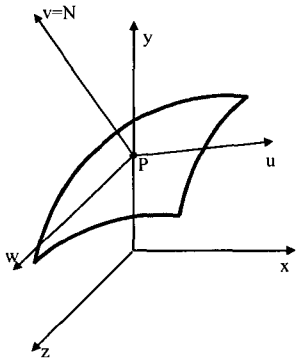


Figura 8.8: El nuevo sistema de referencia $\{u, v, w\}$ está situado sobre el punto $(0, a, 0)$ del sistema $\{x, y, z\}$, y su vector v tiene la misma dirección que la normal a la superficie en dicho punto

Veamos primero cómo son las expresiones para los vectores u, v y w. El vector v coincide con la normal a la superficie en el punto P (expresión 8.1), y por tanto su expresión es

$$v = \frac{1}{M_v}[-b, 1, -c]$$

donde hemos dividido por $M_v = \sqrt{1 + b^2 + c^2}$ para hacerlo unitario.

Como vector w podemos coger cualquier vector perpendicular a v. Tomemos por ejemplo el vector que pertenece al plano yz. Éste será

$$w = \frac{1}{M_w}[0, c, 1]$$

donde $M_w = \sqrt{1 + c^2}$.

Finalmente, el vector u será el resultado del producto vectorial $v \times w$:

$$u = v \times w = \frac{1}{M_v M_w}[1 + c^2, b, -bc]$$

Por lo tanto, las expresiones para el cambio de sistema de coordenadas son:

$$\begin{pmatrix} u & v & w \end{pmatrix} = \begin{pmatrix} x & y-a & z \end{pmatrix} \begin{pmatrix} \frac{1+c^2}{M_v M_w} & -\frac{b}{M_v} & 0 \\ \frac{b}{M_v M_w} & \frac{1}{M_v} & \frac{c}{M_w} \\ -\frac{bc}{M_v M_w} & -\frac{c}{M_v} & \frac{1}{M_w} \end{pmatrix} =$$

$$= \begin{pmatrix} -\dfrac{-xc^2 - x - by + ba + zbc}{M_v M_w} & -\dfrac{xb - y + a + zc}{M_v} & \dfrac{cy - ca + z}{M_w} \end{pmatrix}$$

Lo que necesitamos ahora es resolver el sistema para encontrar la expresión inversa, es decir, las expresiones para $\{x, y, z\}$ en función de las variables $\{u, v, w\}$. Resolviendo el sistema anterior obtenemos las siguientes expresiones

$$\begin{aligned} x &= \frac{M_w}{M_v}u - \frac{b}{M_v}v \\ y &= a + \frac{b}{M_v M_w}u + \frac{1}{M_v}v + \frac{c}{M_w}w \\ z &= -\frac{bc}{M_v M_w}u - \frac{c}{M_v}v + \frac{1}{M_w}w \end{aligned} \tag{8.2}$$

Sustituyendo estas expresiones en la ecuación original del paraboloide obtenemos una nueva ecuación, expresada en el sistema $\{u, v, w\}$. Esta expresión es

$$Av + Buv + Cuw + Dvw + Eu^2 + Fv^2 + Gw^2 = 0$$

donde

$$\begin{aligned} A &= -M_v \\ B &= \frac{2gbc^2 - 2dM_w^2 b + fb^2 c - fM_w^2 c}{M_v^2 M_w} \\ C &= -\frac{2gbc - fM_w^2}{M_v M_w^2} \\ D &= -\frac{2gc + fb}{M_v M_w} \\ E &= \frac{gb^2c^2 + dM_w^4 - fbcM_w^2}{M_v^2 M_w^2} \\ F &= \frac{db^2 + gc^2 + fbc}{M_v^2} \\ G &= \frac{g}{M_w^2} \end{aligned}$$

En la figura 8.9 podemos observar cómo ahora nuestro punto de interés es el $(0, 0, 0)$, y que el vector normal coincide con el eje v, ya que el vector gradiente en cualquier punto es

$$N(u, v, w) = \left[\begin{array}{ccc} Bv + Cw + 2Eu & A + Bu + Dw + 2Fv & Cu + Dv + 2Gw \end{array} \right]$$

y en el punto $(0, 0, 0)$ es justamente $N = [0, A, 0]$.

Cálculo de los valores de curvatura principales

A continuación vamos a barrer el plano uw con un ángulo θ para obtener la curva intersección entre la superficie y todos los planos normales al plano uw, como se aprecia en la figura 8.10.

Sea t la dirección paralela al plano uw que forma un ángulo θ con el eje w, es decir

$$\begin{aligned} w &= t\cos\theta \\ u &= t\sin\theta \end{aligned}$$

Calculemos la expresión para la curva intersección entre el plano tv y el paraboloide:

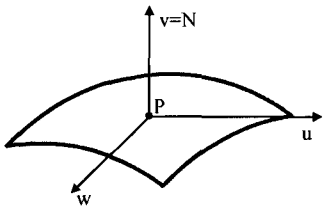


Figura 8.9: En el sistema $\{u, v, w\}$, el origen está sobre el paraboloide, y el vector v coincide con su vector normal

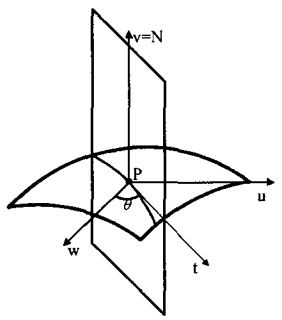


Figura 8.10:

$$Av + Btv\sin\theta + Ct^2\sin\theta\cos\theta + Dvt\cos\theta + Et^2\sin^2\theta + Fv^2 + Gt^2\cos^2\theta = H(t, v, \theta) = 0$$

Esto significa que, para cada valor de θ entre 0 y 2π tenemos una curva 2D en forma implícita, $H(t, v) = 0$, a la cual queremos calcular su curvatura en el punto $(0, 0)$. La expresión para la curvatura de una curva implicita viene dada por la fórmula

$$K(\theta, t, v) = \frac{H_{tt}H_v^2 - 2H_{tv}H_tH_v + H_{vv}H_t^2}{\left(H_t^2 + H_v^2\right)^{3/2}}$$

Evaluando dicha expresión para el punto $(0, 0)$ obtenemos que

$$K(\theta) = \frac{2}{A}\left(C\sin\theta\cos\theta + E\sin^2\theta + G\cos^2\theta\right) \tag{8.3}$$

Ahora sólo falta encontrar para qué valores de θ se encuentran los máximos y los mínimos de $K(\theta)$. Derivando e igualando a 0 obtenemos que los extremos se encuentran para los ángulos

$$\alpha = \frac{1}{2}\arctan\left(\frac{C}{G - E}\right)$$

donde aparecen 4 soluciones en el intervalo $[-\pi, \pi]$, todas ellas igualmente espaciadas en el plano uw (lo que en realidad significa que son dos direcciones diferentes, perpendiculares entre sí). Por tanto, llamando α a cualquiera de las 4 posibles soluciones de la arcotangente, podemos afirmar que las curvaturas principales del paraboloide son

$$\begin{aligned} K_1 &= \frac{2}{A}\left(C\sin\alpha\cos\alpha + E\sin^2\alpha + G\cos^2\alpha\right) \\ K_2 &= \frac{2}{A}\left(-C\cos\alpha\sin\alpha + E\cos^2\alpha + G\sin^2\alpha\right) \end{aligned}$$

Cálculo de las direcciones de curvatura principales

Además de conocer la magnitud de las curvaturas máxima y mínima, también es interesante conocer en qué direcciones se producen, pero medidas en el sistema original $\{x, y, z\}$. En el sistema $\{u, v, w\}$ sus expresiones son bastante sencillas

$$V_1 = [\sin\alpha, 0, \cos\alpha]$$

Ahora hay que convertirlo al sistema $\{x, y, z\}$. Primero consideremos que nuestro vector viene dado por los puntos $P_1 = (0, 0, 0)$ y $P_2 = (\sin\alpha, 0, \cos\alpha)$, como se ve en la figura 8.11, y transformemos estos puntos al sistema original.

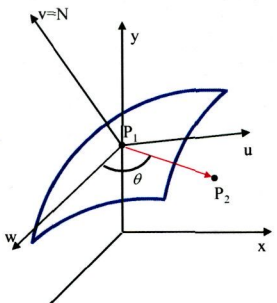


Figura 8.11:

Ya sabemos que $P_1 = (0, a, 0)$. Vayamos con P_2, y para ello usemos las ecuaciones que nos pasan del sistema $\{u, v, w\}$ al $\{x, y, z\}$ (ecuacion 8.2). La expresión final para el punto es

$$P_2 = \left(\frac{M_w}{M_v}\sin\alpha, \frac{aM_vM_w + b\sin\alpha + cM_v\cos\alpha}{M_vM_w}, -\frac{bc\sin\alpha - M_v\cos\alpha}{M_vM_w}\right)$$

Por lo tanto, la expresión del vector V_1 en el sistema $\{x, y, z\}$ es

$$V_1 = \left[\ M_w^2\sin\alpha, \quad b\sin\alpha + cM_v\cos\alpha, \quad M_v\cos\alpha - bc\sin\alpha\ \right]$$

cuyo módulo es igual a M_vM_w. Con esto podemos afirmar que el vector normalizado para una de las curvaturas extremas es

$$V_1 = \frac{1}{M_vM_w}\left[\ M_w^2\sin\alpha, \quad b\sin\alpha + cM_v\cos\alpha, \quad M_v\cos\alpha - bc\sin\alpha\ \right] \tag{8.4}$$

El otro vector se producirá en $\alpha + \pi/2$, con lo cual su expresión será

$$V_2 = \frac{1}{M_v M_w} \left[\; M_w^2 \cos\alpha, \quad b\cos\alpha - cM_v \sin\alpha, \quad -M_v \sin\alpha - bc\cos\alpha \; \right] \tag{8.5}$$

Eliminación de la función arcotangente

Hemos visto que para calcular el ángulo α necesitamos usar la función arcotangente. Computacionalmente es mejor utilizar raíces cuadradas, debido a los errores en valores muy grandes del argumento. Podemos eliminar de las expresiones anteriores todas las funciones trigonómetricas haciendo el siguiente desarrollo. En primer lugar es fácil deducir que la expresión para la curvatura (ecuación 8.3) puede ponerse también como

$$K(\theta) = \frac{1}{A}\left(E + G + C\sin 2\theta + (G - E)\cos 2\theta\right)$$

usando las fórmulas trigonométricas para los ángulos dobles. Por lo tanto, una de las curvaturas principales, $K(\alpha)$, podemos expresarla como

$$K(\alpha) = \frac{1}{A}\left(E + G + C\sin 2\alpha + (G - E)\cos 2\alpha\right) = \frac{1}{A}\left(E + G + R\right)$$

donde $R = \sqrt{C^2 + (G-E)^2}$ y donde se ha usado el hecho de que

$$\begin{aligned} \sin 2\alpha &= \frac{C}{R} \\ \cos 2\alpha &= \frac{G - E}{R} \end{aligned}$$

La otra curvatura principal será

$$K(\alpha + \pi/2) = \frac{1}{A}\left(E + G - R\right)$$

Teniendo en cuenta que $A < 0$, podemos afirmar que la curvatura máxima será siempre $K_{\max} = K(\alpha + \pi/2)$ y la mínima será $K_{\min} = K(\alpha)$.

Vectores dirección Para eliminar las funciones trigonométricas de las expresiones de los vectores V_1 y V_2 (ecuaciones 8.4 y 8.5) usaremos el hecho de que

$$\begin{aligned} \sin\alpha &= \sqrt{\frac{R - (G-E)}{2R}} \\ \cos\alpha &= \sqrt{\frac{R + (G-E)}{2R}} \end{aligned}$$

De esta manera, las expresiones quedan como sigue:

$$\begin{aligned} V_{\min} = V_1 &= \frac{1}{M_v M_w \sqrt{2R}} \left[\; M_w^2 S, \quad bS + cM_v T, \quad M_v T - bcS \; \right] \\ V_{\max} = V_2 &= \frac{1}{M_v M_w \sqrt{2R}} \left[\; M_w^2 T, \quad bT - cM_v S, \quad -M_v S - bcT \; \right] \end{aligned}$$

donde $S = \sqrt{R - (G-E)}$ y $T = \sqrt{R + (G-E)}$. Para que funcione en todos los casos, los signos de estas dos raíces son importantes. Es fácil demostrar que S debe tener el mismo signo que C, y T debe ser siempre positivo.

Caso $F_y < 0$

Cuando la derivada parcial en y es negativa significa que la intensidad bajo la superficie es menor que sobre ella, es decir, $A < B$. En este caso, y al igual que ocurría en 2D, los signos de las curvaturas se ven invertidos. La razón era sencilla. Consideremos la imagen de una esfera maciza con una intensidad en el interior y otra en el exterior, como se ve en la figura 8.12. Nos interesa que el valor de la curvatura sea el mismo en todos los voxels de su superficie. Sin embargo, las zonas indicadas en la figura, cuando estimamos el paraboloide $y = f(x, z)$ que mejor aproxima el borde, tienen curvaturas de distinto signo.

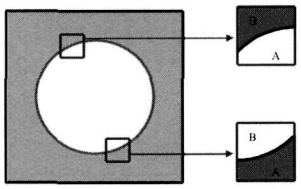


Figura 8.12: En ambas zonas la curvatura debe ser idéntica

Por ello, las expresiones anteriores son válidas para el caso $F_y > 0$. En caso contrario, habrá que invertir el signo de ambas curvaturas, con lo cual la máxima sera la que antes era la mínima y viceversa. De igual manera, las direcciones habrá que intercambiarlas entre sí. Estos resultados se resumen en el siguiente lema:

Lema 8.1 *Sea el paraboloide dado por la ecuación* $y = a + bx + cz + dx^2 + fxz + gz^2$. *Las curvaturas principales en el punto* $(0, a, 0)$ *son*

$$\begin{aligned} K_{\min} &= \min\{K_1, K_2\} \\ K_{\max} &= \max\{K_1, K_2\} \end{aligned}$$

donde

$$\begin{aligned} K_1 &= \frac{n}{A}(E + G + R) \\ K_2 &= \frac{n}{A}(E + G - R) \\ R &= \sqrt{C^2 + (G - E)^2} \\ n &= \begin{cases} 1 & \text{si } F_y \geq 0 \\ -1 & \text{si } F_y < 0 \end{cases} \\ A &= -M_v \\ C &= -\frac{2gbc - fM_w^2}{M_v M_w^2} \\ E &= \frac{gb^2c^2 + dM_w^4 - fbcM_w^2}{M_v^2 M_w^2} \\ G &= \frac{g}{M_w^2} \\ M_v &= \sqrt{1 + b^2 + c^2} \\ M_w &= \sqrt{1 + c^2} \end{aligned}$$

y las direcciones en las que se producen vienen dadas por los vectores

$$V_1 = \frac{1}{M_v M_w \sqrt{2R}} \left[M_w^2 S, \quad bS + cM_v T, \quad M_v T - bcS \right]$$
$$V_2 = \frac{1}{M_v M_w \sqrt{2R}} \left[M_w^2 T, \quad bT - cM_v S, \quad -M_v S - bcT \right]$$

donde

$$S = p\sqrt{R-(G-E)}$$
$$T = \sqrt{R+(G-E)}$$
$$p = \begin{cases} 1 & \text{si } C \geq 0 \\ -1 & \text{si } C < 0 \end{cases}$$

8.2.4 Lema final

Todos los resultados de esta sección se resumen en el siguiente lema:

Lema 8.2 *Sea $F(i,j,k)$ una imagen 3D por cuyo voxel $(0,0,0)$ pasa un contorno que separa dos zonas de intensidad A (debajo del contorno) y B (encima del contorno), y que cumple que $|F_y|\,h_y > |F_x|\,h_x, |F_z|\,h_z$. Si suponemos que en la vecindad de ese voxel, el contorno se comporta como una superficie de segundo grado, podemos estimar los parámetros de dicho contorno (posición, vector normal y curvaturas) calculando previamente el paraboloide que mejor se aproxima a dicho borde. Los pasos son los siguientes:*

a) obtenemos las sumas de las columnas de voxels de la vecindad, de la siguiente manera:

$$S_1 = \sum_{j=-3}^{3} F_{1,j,0} \qquad S_2 = \sum_{j=-3}^{3} F_{0,j,-1} \qquad S_3 = \sum_{j=-3}^{3} F_{0,j,0}$$
$$S_4 = \sum_{j=-3}^{3} F_{0,j,1} \qquad S_5 = \sum_{j=-3}^{3} F_{-1,j,0} \qquad S_6 = \sum_{j=-3}^{3} (F_{1,j,-m} + F_{-1,j,m})$$

donde

$$m = \begin{cases} 1 & \text{si } F_x F_z \geq 0 \\ -1 & \text{si } F_x F_z < 0 \end{cases}$$

b) obtenemos también una estimación para los valores de intensidad a ambos lados del borde

$$A = \frac{1}{9}\Big(\begin{array}{c} F_{-\alpha,-3,\beta} + F_{-\alpha,-3,0} + F_{-\alpha,-3,-\beta} + F_{0,-3,0} + F_{0,-3,-\beta} + F_{\alpha,-3,-\beta} + \\ + F_{-\alpha,-2,0} + F_{-\alpha,-2,-\beta} + F_{0,-2,-\beta} + F_{-\alpha,-1,-\beta} \end{array} \Big)$$
$$B = \frac{1}{9}(F_{\alpha,3,-\beta} + F_{\alpha,3,0} + F_{\alpha,3,\beta} + F_{0,3,0} + F_{0,3,\beta} + F_{-\alpha,3,\beta} + F_{\alpha,2,0} + F_{\alpha,2,\beta} + F_{0,2,\beta} + F_{\alpha,1,\beta})$$

donde

$$\alpha = \begin{cases} 1 & \text{si } F_x F_y \geq 0 \\ -1 & \text{si } F_x F_y < 0 \end{cases} \qquad \beta = \begin{cases} 1 & \text{si } F_z F_y \geq 0 \\ -1 & \text{si } F_z F_y < 0 \end{cases}$$

c) a continuación detectamos los coeficientes del paraboloide $y = a + bx + cz + dx^2 + fxz + gz^2$ como sigue:

$$a = \frac{h_y}{24(A-B)}(28S_3 - S_1 - S_2 - S_4 - S_5 - 84A - 84B)$$
$$b = \frac{h_y}{2h_x(A-B)}(S_1 - S_5)$$
$$c = \frac{h_y}{2h_z(A-B)}(S_4 - S_2)$$
$$d = \frac{h_y}{2(A-B)h_x^2}(S_1 + S_5 - 2S_3)$$
$$f = \frac{mh_y}{2(A-B)h_x h_z}(S_1 + S_2 + S_4 + S_5 - S_6 - 2S_3)$$
$$g = \frac{h_y}{2(A-B)h_z^2}(S_2 + S_4 - 2S_3)$$

d) la posición y el vector normal se calcula a partir de la expresión del paraboloide

$$\begin{aligned} \text{corte con la vertical central del voxel} &: (0, a, 0) \\ \text{vector normal} &: N = \frac{A - B}{\sqrt{1 + b^2 + c^2}} \begin{bmatrix} b & -1 & c \end{bmatrix} \end{aligned}$$

e) las curvaturas principales (magnitud y dirección) se calculan con el lema 8.1

Hay que tener en cuenta que el anterior lema sólo es aplicable para el caso $|F_y|\, h_y > |F_x|\, h_x, |F_z|\, h_z$. Cuando esto no ocurra, el esquema será idéntico, pero intercambiando las variables entre sí. De esta manera, si la parcial mayor fuera por ejemplo la de la dirección x, es decir, se cumpliese que $|F_x|\, h_x > |F_y|\, h_y, |F_z|\, h_z$, el lema a aplicar sería equivalente pero realizando las sumas por filas en lugar de por columnas, e intercambiando los índices y las variables x e y, así como las componentes de los vectores.

8.3 Algoritmo para la localización de contornos de segundo orden

Al igual que en el método lineal, en primer lugar habrá que detectar qué voxels son etiquetados como borde. Para ello, obtendremos las tres imágenes de parciales, F_x, F_y y F_z, y nos centraremos en aquellos voxels (i, j, k) donde

$$F_x^2(i,j,k) + F_y^2(i,j,k) + F_z^2(i,j,k) > \delta^2$$

donde δ es un umbral que habrá que prefijar, y que además cumplan que

$$|F_y(i,j-2,k)| < |F_y(i,j-1,k)| < |F_y(i,j,k)| > |F_y(i,j+1,k)| > |F_y(i,j+2,k)|$$

en el caso en que $|F_y(i,j,k)|\, h_y > |F_x(i,j,k)|\, h_x, |F_z(i,j,k)|\, h_z$, o

$$|F_x(i-2,j,k)| < |F_x(i-1,j,k)| < |F_x(i,j,k)| > |F_x(i+1,j,k)| > |F_x(i+2,j,k)|$$

en el caso en que $|F_x(i,j,k)|\, h_x > |F_y(i,j,k)|\, h_y, |F_z(i,j,k)|\, h_z$, o bien que

$$|F_z(i,j,k-2)| < |F_z(i,j,k-1)| < |F_z(i,j,k)| > |F_z(i,j,k+1)| > |F_z(i,j,k+2)|$$

en el caso en que $|F_z(i,j,k)|\, h_z > |F_x(i,j,k)|\, h_x, |F_y(i,j,k)|\, h_y$.

Una vez tenemos los voxels identificados, aplicaremos alrededor de cada uno el lema 8.2 para obtener los parámetros del contorno (posición, vector normal, salto de intensidad y curvaturas principales). Esto significa que, para cada voxel borde, el método usará los valores de un entorno 3x7x3 centrado en dicho voxel, cuya orientación variará en función de qué parcial es mayor.

Hay cinco condiciones que deben cumplirse dentro de dicho entorno, o al menos acercarse lo más posible, para que la situación alrededor de esos voxels sea lo más parecido a la situación ideal sobre la que se ha desarrollado el método:

1) Dentro de ese entorno, el contorno puede ser aproximado como una superficie de grado 2.
2) Dentro de ese entorno, no debe aparecer ningún otro contorno diferente.
3) Dentro de ese entorno, la intensidad a ambos lados del contorno debe ser lo más homogénea posible.
4) La superficie debe atravesar el entorno lateralmente, sin tocar nunca las caras inferior ni superior.
5) La imagen debe poseer el menor ruido posible

Lógicamente, es difícil que las cinco condiciones se cumplan exhaustivamente en todos los voxels borde, pero también es bastante probable que se cumplan en un grado alto. La primera y la cuarta condición fallarán en zonas donde la curvatura del contorno sea muy alta. La segunda fallará en zonas donde haya dos superficies

muy cercanas entre sí. La tercera fallará en los bordes de zonas con textura poco homogénea. En todos los demás casos, las condiciones se cumplirán.

En cuanto a la quinta condición, en el presente capítulo seguiremos trabajando con imágenes sintéticas, y ya en el próximo estudiaremos como evitar el error producido por el ruido.

El pseudo-código de nuestro algoritmo podría ser el siguiente:

Funcion ContornosCurvos3D(delta)

```
Calcular Fx, Fy, Fz
Para todos los voxels (i,j,k) de la imagen
   Si Fx(i,j,k)*Fx(i,j,k)+Fy(i,j,k)*Fy(i,j,k)+Fz(i,j,k)*Fz(i,j,k) < delta*delta Continuar
   Si (|Fx(i,j,k)|hx>|Fy(i,j,k)|hy) y (|Fx(i,j,k)|hx>|Fz(i,j,k)|hz) Entonces
      Si |Fx(i,j,k)| no es maximo en su fila Continuar
      Aplicar lema 8.2 con la variable x como direccion principal
   Fin Si
   Si (|Fy(i,j,k)|hy>|Fx(i,j,k)|hx) y (|Fy(i,j,k)|hy>|Fz(i,j,k)|hz) Entonces
      Si |Fy(i,j,k)| no es maximo en su columna Continuar
      Aplicar lema 8.2 con la variable y como direccion principal
   Fin Si
   Si (|Fz(i,j,k)|hz>|Fx(i,j,k)|hx) y (|Fz(i,j,k)|hz>|Fy(i,j,k)|hy) Entonces
      Si |Fz(i,j,k)| no es maximo en su fila Continuar
      Aplicar lema 8.2 con la variable z como direccion principal
   Fin Si
Fin Para
```

8.4 Ejemplos

Vamos a aplicar el método a distintas superficies geométricas generadas sintéticamente, y compararemos los resultados con una de las técnicas tradicionales, como la propuesta por Álvarez (ver sección 8.1.1).

8.4.1 Esferas

Comencemos por la imagen de la misma esfera que generamos en el capítulo anterior, de radio 20 voxels. Si rehacemos la tabla del capítulo anterior (tabla 7.2) añadiendo el nuevo método propuesto de segundo orden (ver tabla 8.1) podemos ver como la estimación del salto de intensidad y de la dirección del vector normal coincide en los dos métodos propuestos (lineal y cuadrático). Esto es lógico porque para estimar la intensidad usamos voxels similares, y para deducir el vector normal no se necesitan las derivadas segundas.

La diferencia está en la estimación de la posición sub-voxel, que ahora es bastante más precisa. ¿Por qué no es del todo exacta? La respuesta es que existe una pequeña diferencia entre un trozo de esfera y un trozo de paraboloide, que es lo que realmente trata de detectar nuestro método, y de ahí el error. Es la misma razón que ocurría en 2D cuando tratábamos de detectar una circunferencia. En el último capítulo abordaremos más en profundidad este problema.

En la columna izquierda de la figura 8.13 se muestra para cada voxel de la superficie el error cometido en el cálculo de la posición subvoxel para los métodos propuestos cuadrático y lineal. Vemos que el error ha disminuido en todos los voxels.

En cuanto a las curvaturas, en la tabla 8.2 se muestran los resultados del método propuesto, comparados con los del método tradicional. Teóricamente existe una única curvatura sobre la superficie de la esfera, igual a

	Error salto de intensidad				Error dirección normal				Error posición			
Método	Med.	Desv.	Mín.	Max.	Med.	Desv.	Mín.	Max.	Med.	Desv.	Mín.	Max.
Prop. cuad.	0.00	0.00	0.00	0.00	0.15	0.08	0.00	0.53	0.25	0.20	0.00	0.83
Prop. lineal	0.00	0.00	0.00	0.00	0.15	0.08	0.00	0.53	0.65	0.32	0.01	1.50
Tradicional	6.98	6.00	0.00	21.8	2.98	1.08	0.00	4.85	–	–	–	–

Tabla 8.1: Errores cometidos por cada método al obtener el contorno de una esfera de radio 20

	Curvatura mínima				Curvatura máxima			
Método	Media	Desv.	Mín.	Max.	Media	Desv.	Mín.	Max.
Propuesto	$0.047(R = 21.1)$	0.005	0.029	0.054	$0.053(R = 19.0)$	0.005	0.048	0.070
Tradicional	$0.020(R = 51.3)$	0.014	−0.024	0.036	$0.040(R = 24.8)$	0.026	−0.037	0.093

Tabla 8.2: Errores cometidos por cada método al obtener el contorno de una esfera de radio 20

la inversa de su radio ($1/20 = 0.05$). Sin embargo, numéricamente es normal que salgan dos valores ligeramente diferentes, y es por eso por lo que la información se muestra en dos columnas.

Como puede apreciarse, el método propuesto consigue un error medio bastante pequeño, de tres milésimas menos para la curvatura mínima y tres más para la máxima. Esto significa que el radio de curvatura mínimo medio para todos los voxels borde de la esfera es de 19.0 unidades, mientras que el radio máximo medio es de 21.1. Téngase en cuenta que la diferencia entre una pequeña porción de esfera de radio 20 y otra de 21 es prácticamente inapreciable. Comparando con el método tradicional, el error medio es bastante mayor (radios medios de 51 y 25 unidades).

Otro detalle es la desviación típica del valor de curvatura. Con el método propuesto, la oscilación entre todos los voxels es bastante pequeña (0.005), lo cual significa que en la gran mayoría de los voxels de la superficie la medida de ambas curvaturas es bastante acertada. Sin embargo, con el método tradicional, la desviación es bastante más alta, llegando incluso a haber voxels con curvatura de distinto signo (−0.037), es decir, voxels cuyo radio de curvatura estimado es de 27 unidades pero de forma cóncava en lugar de convexa.

En la figura 8.13 (columnas central y derecha) se muestra el error cometido en la estimación de la curvatura mínima y máxima respectivamente. Mientras que con el método propuesto el error en cada voxel es menor y está repartido de forma más homogénea sobre la superficie, con el método tradicional el error es bastante alto en algunas zonas concretas.

En cuanto a las direcciones de las curvaturas, al tratarse de una esfera, se trata de información irrelevante, ya que la curvatura es la misma en cualquier dirección. Es por eso que no hemos representado una gráfica sobre las direcciones. Lo haremos con el siguiente objeto geométrico.

8.4.2 Cilindros

Tomemos ahora la imagen de un cilindro de radio 15, orientado a lo largo del eje z, como el de la figura 8.14a. Cualquier punto de la superficie del cilindro tendrá siempre dos valores de curvatura: 0 en la dirección z, y $1/15$ en la dirección tangente sobre el plano xy.

En la tabla 8.3 se muestra el resultado de comparar nuestro método cuadrático propuesto con el tradicional. Las diferencias importantes son en la estimación del salto de intensidad (error nulo en el propuesto frente a más de 10 unidades en algunos casos con el tradicional), la dirección del vector normal (0.15 grados de media frente a casi 2 grados y medio en el tradicional) y la curvatura máxima (radio medio de 15.0 unidades frente a 23.6 unidades con el tradicional, con mucha más desviación típica, llegando a haber curvaturas negativas). En cuanto a la curvatura mínima y las direcciones de curvatura, ambos métodos son bastante precisos.

En la figura 8.15 se muestra el error cometido en cada voxel de la superficie. Vemos cómo el método propuesto

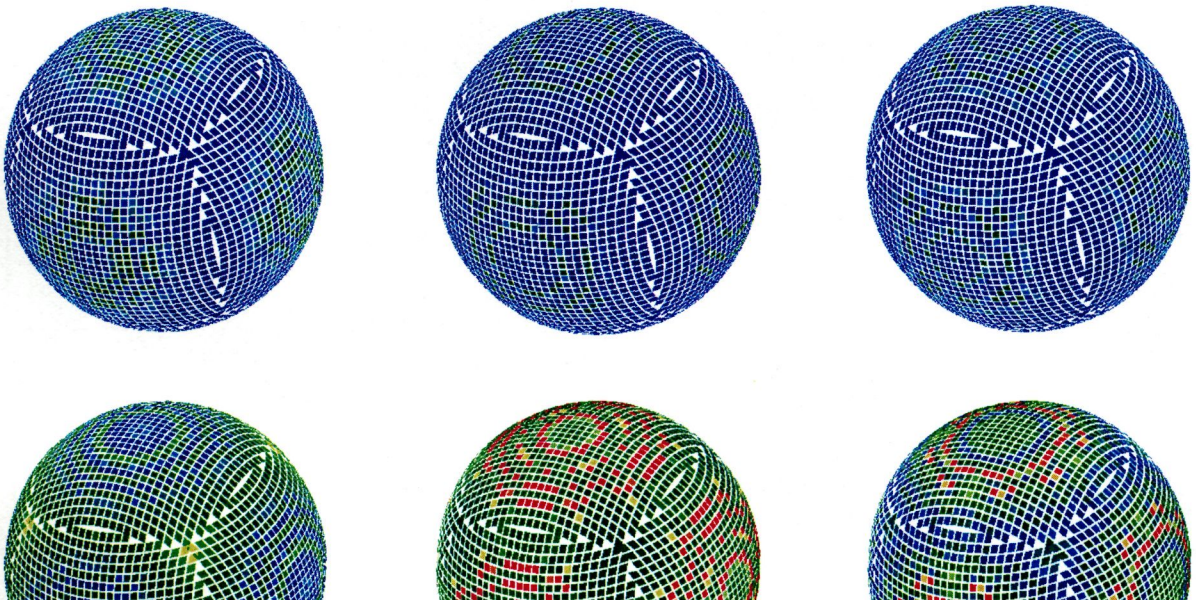


Figura 8.13: Error cometido al detectar el contorno de una esfera de radio 20 utilizando el método propuesto de segundo orden (fila superior) y otro método diferente (fila inferior). De izquierda a derecha, error cometido en la estimación de la posición (comparando con el método propuesto lineal), y de las curvaturas mínima y máxima (comparando con el método de Luis Álvarez)

se comporta bastante bien para todas las magnitudes. Por el contrario, usando el método tradicional obtenemos muchos voxels con un error en cuanto al salto de intensidad de más de 5 unidades, teniendo además la mayoría de ellos un error con respecto a la dirección del vector normal de más de tres grados. Sin embargo, en cuanto a las direcciones de curvatura parece tan preciso como el nuestro. Pero esto ha sido realmente porque hemos usado un cilindro perfectamente orientado a lo largo de uno de los ejes principales. Si repetimos las pruebas pero utilizando un cilindro orientado en una dirección arbitraria, los errores serán mayores con el método tradicional.

En la tabla 8.4 se muestra el resultado de comparar nuestro método cuadrático propuesto con el tradicional, aplicado a un cilindro de radio 15 orientado en la dirección dada por el vector $[3, 5, 7]$, como el mostrado en la figura 8.14b. Ahora las diferencias son importantes en todas las magnitudes, sobre todo en las direcciones de curvatura, donde el método tradicional puede llegar en algunos voxels a los 90 grados de error. La razón de este error tan alto es la siguiente.

Aplicando el método propuesto, la magnitud de la curvatura mínima oscila en el intervalo $[-0.019, 0.021]$, y la curvatura máxima en $[0.060, 0.074]$. No existe intersección entre ambos intervalos. Sin embargo, con el método tradicional, los intervalos de ambas curvaturas son $[-0.033, 0.035]$ para la mínima y $[-0.065, 0.096]$

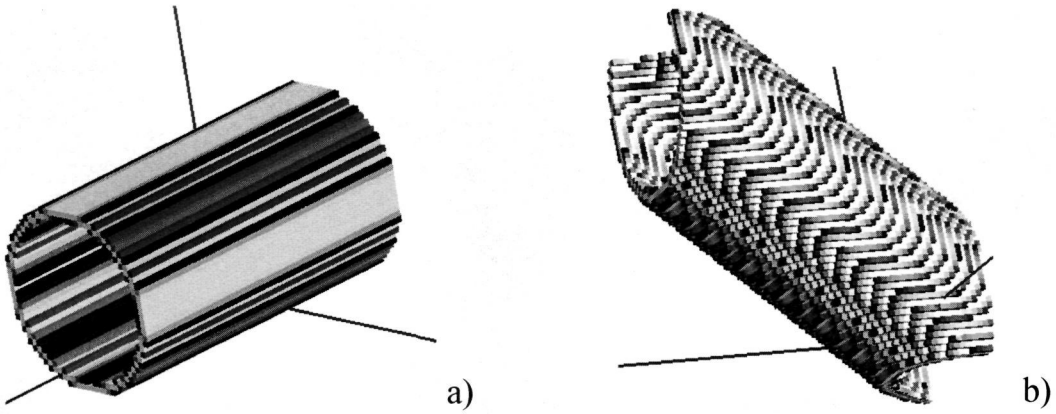


Figura 8.14: a) Cilindro de radio 15 orientado en el eje z; b) Igual pero orientado en la dirección $[3, 5, 7]$

	Error salto de intensidad				**Error dirección normal**				**Error posición**			
Método	Med.	Des.	Mín.	Max.	Med.	Des.	Mín.	Max.	Med.	Des.	Mín.	Max.
Prop.	0.00	0.00	0.00	0.00	0.15	0.08	0.00	0.26	0.29	0.21	0.02	0.65
Tradic.	4.12	3.76	0.00	10.9	2.46	1.22	0.00	3.93	–	–	–	–
	Curvatura mínima					**Curvatura máxima**						
	Rad.	Med.	Des.	Mín.	Máx.	Rad.	Med.	Des.	Mín.	Máx.		
Prop.	∞	0.00	0.00	0.00	0.00	15.0	0.067	0.008	0.055	0.080		
Tradic.	∞	0.00	0.00	0.00	0.00	23.6	0.042	0.034	-0.012	0.087		
	Error dirección curvatura mínima					**Error dirección curvatura máxima**						
		Med.	Des.	Mín.	Máx.		Med.	Des.	Mín.	Máx.		
Prop.		0.00	0.00	0.00	0.00		0.00	0.00	0.00	0.01		
Tradic.		0.00	0.01	0.00	0.01		0.01	0.00	0.00	0.01		

Tabla 8.3: Errores cometidos por cada método al obtener el contorno de un cilindro de radio 15

para la máxima. La intersección es muy grande entre ambos. Esto significa que en algunos voxels la curvatura mínima sea superior a la máxima, provocando que las direcciones de curvatura resulten intercambiadas entre sí, dando lugar a errores cercanos a los 90 grados.

En la figura 8.16 se muestra el error cometido en cada voxel de la superficie. De nuevo el método propuesto se comporta bastante bien en todas las magnitudes. Por el contrario, el método tradicional comete un error alto en el cálculo del salto de intensidad (más de 5 unidades en la gran mayoría de los voxels), en la dirección del vector normal (más de 3 grados en casi todos los voxels) y en las direcciones de curvatura (más de 15 grados de error en la mayoría de la superficie).

Estos errores, además, son bastante caóticos cuando se mira en detalle una zona contigua de superficie, tal y como se aprecia en la figura 8.17, donde en cada voxel se ha usado una pequeña línea negra para mostrar la dirección de curvatura estimada.

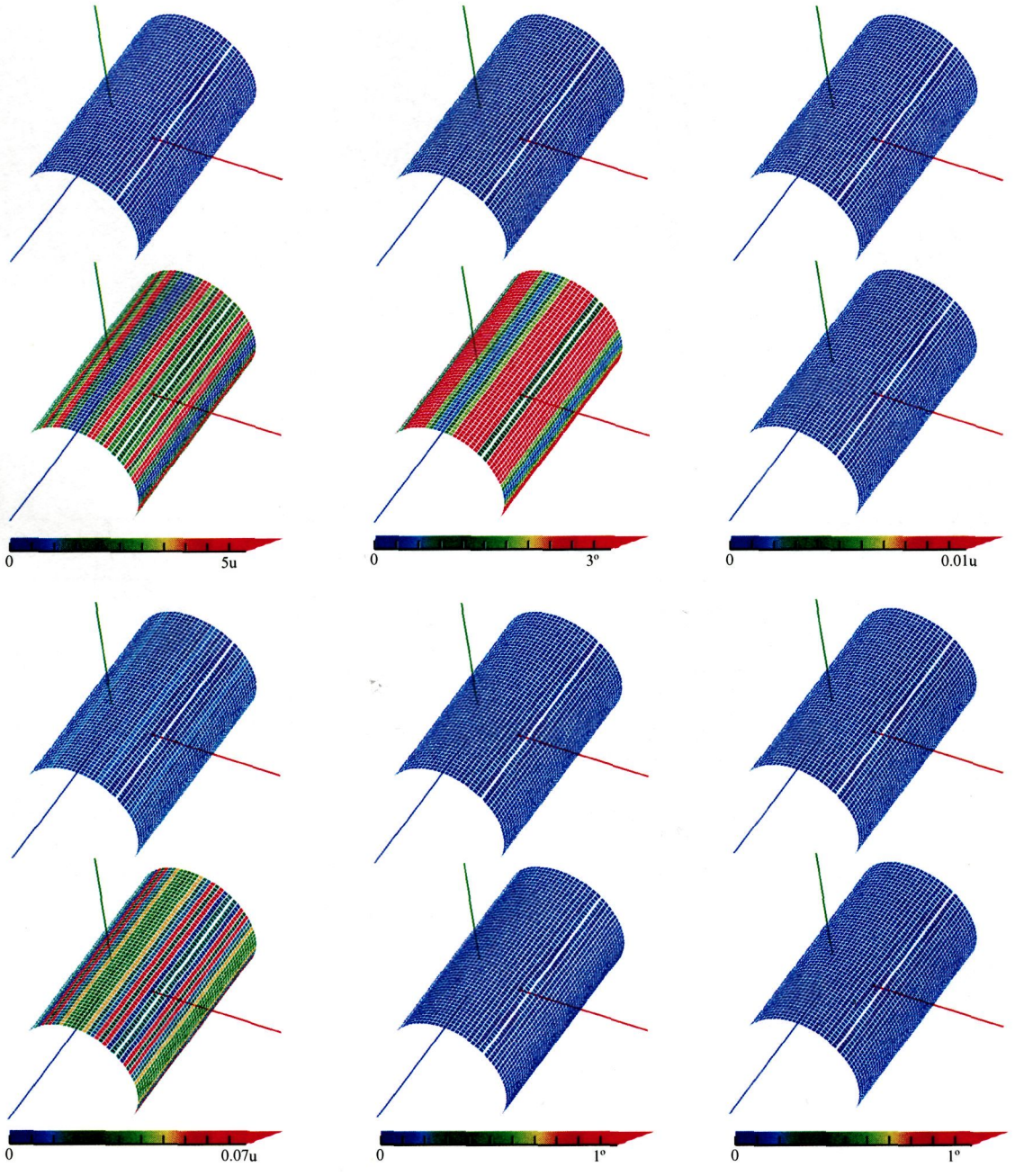


Figura 8.15: Error cometido al detectar el contorno de un cilindro de radio 15 orientado en el eje z utilizando el método propuesto (filas 1 y 3) y el método tradicional (filas 2 y 4). De izquierda a derecha, y de arriba a abajo: error cometido en la estimación del salto de intensidad, dirección normal, curvatura mínima y máxima, y direcciones de curvatura mínima y máxima

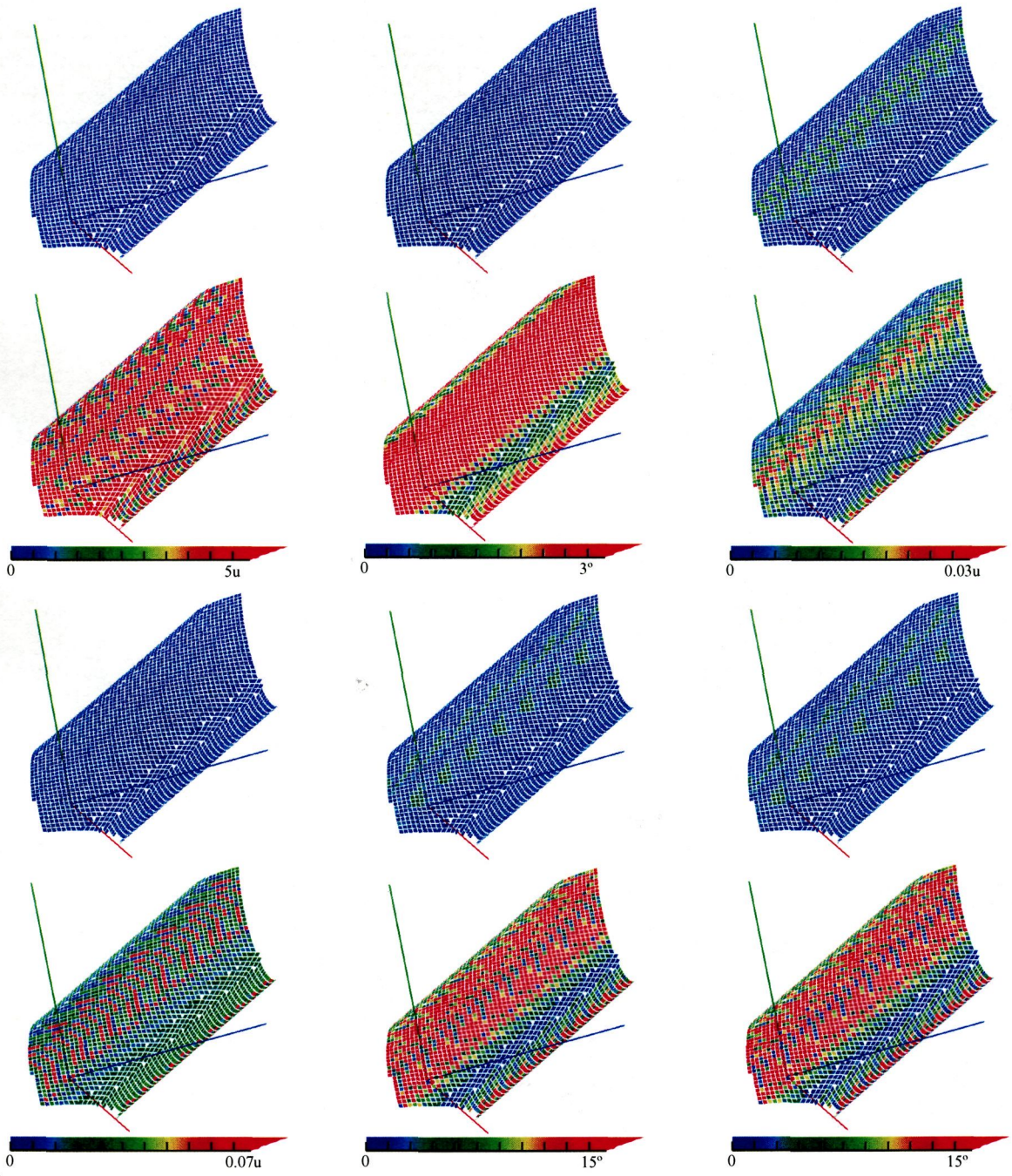


Figura 8.16: Error cometido al detectar el contorno de un cilindro de radio 15 orientado en la dirección $[3, 5, 7]$ utilizando el método propuesto (filas 1 y 3) y el método tradicional (filas 2 y 4). De izquierda a derecha, y de arriba a abajo: error cometido en la estimación del salto de intensidad, dirección normal, curvatura mínima y máxima, y direcciones de curvatura mínima y máxima

	Error salto de intensidad				**Error dirección normal**				**Error posición**			
Método	Med.	Des.	Mín.	Max.	Med.	Des.	Mín.	Max.	Med.	Des.	Mín.	Max.
Prop.	0.00	0.00	0.00	0.00	0.11	0.07	0.00	0.32	0.19	0.17	0.00	0.69
Tradic.	8.07	5.96	0.00	25.2	3.12	1.04	0.37	4.76	–	–	–	–
	Curvatura mínima					**Curvatura máxima**						
	Rad.	Med.	Des.	Mín.	Máx.	Rad.	Med.	Des.	Mín.	Máx.		
Prop.	∞	0.000	0.006	−0.019	0.021	14.9	0.067	0.002	0.060	0.074		
Tradic.	160.1	0.006	0.015	−0.033	0.035	29.7	0.034	0.035	−0.065	0.096		
	Error dirección curvatura mínima					**Error dirección curvatura máxima**						
		Med.	Des.	Mín.	Máx.		Med.	Des.	Mín.	Máx.		
Prop.		1.26	1.36	0.00	5.87		1.25	1.36	0.00	5.87		
Tradic.		22.97	24.62	0.78	89.8		22.55	24.89	0.04	89.8		

Tabla 8.4: Errores cometidos por cada método al obtener el contorno de un cilindro de radio 15 orientado en la dirección [3,5,7]

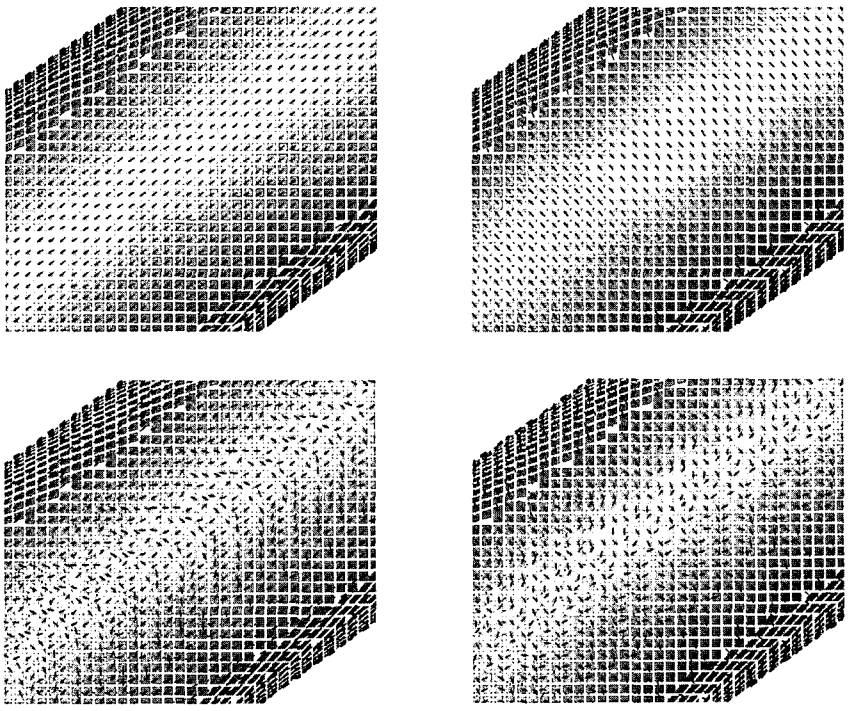


Figura 8.17: Error cometido al detectar las direcciones de curvatura mínima (columna izquierda) y máxima (columna derecha) sobre la imagen de un cilindro de radio 15 orientado en la dirección $[3, 5, 7]$ utilizando el método propuesto (fila superior) y el método tradicional (fila inferior)

8.4.3 Toroides

Por último vamos a hacer una prueba con un toroide sintético. La ecuación de un toroide paralelo al plano xy es

$$\left(R - \sqrt{x^2 + y^2}\right)^2 + z^2 = r^2$$

donde R y r son los radios mayor y menor de la figura, $R > r$, tal y como se ve en la figura 8.18.

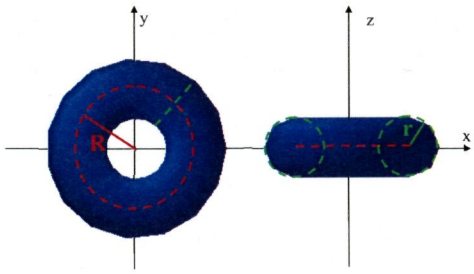


Figura 8.18: Toroide paralelo al plano xy

La curvatura máxima en toda su superficie es $1/r$, mientras que la curvatura mínima se encuentra en el interior del intervalo

$$K_{\min} \in \left[-\frac{1}{R-r}, \frac{1}{R+r}\right]$$

Así tenemos que la curvatura mínima es negativa en los puntos de la circunferencia interior, cero en los puntos con valores extremos en z, y positiva en el anillo exterior. Esta característica hace del toroide una de las figuras más completas en cuanto a configuraciones posibles de la curvatura. Es por ello que, cuando trabajemos con imágenes angiográficas en capítulos posteriores, un trozo de toroide será la forma idónea para simular un trozo de vaso sanguíneo.

Tomemos para realizar las pruebas numéricas un toroide con $R = 30$ y $r = 10$, como el de la figura 8.19a y apliquémosle el método propuesto y el tradicional para detectar los contornos. Vamos a centrarnos además en la información de la curvatura, ya que el resto de magnitudes se comporta de forma similar al de las figuras de las secciones anteriores.

En la columna izquierda de la figura 8.20 se aprecia el error cometido en la estimación de la curvatura máxima, la cual debería ser constante e igual a 0.1. Vemos como el método propuesto es bastante homogéneo en todos sus voxels, mientras que el tradicional comete bastante error.

En la columna derecha se muestra la curvatura mínima, pero el color de cada plano no representa el error en sí, sino el verdadero valor de la curvatura obtenida, la cual teóricamente debería oscilar entre $-1/20$ y $1/40$, es decir entre -0.05 y 0.025. Podemos apreciar como efectivamente, con el método propuesto, el color de los voxels varía suavemente entre el azul (-0.05) en los puntos más interiores del toroide hasta llegar al rojo ($+0.025$) en los puntos exteriores, pasando por el amarillo (0.0) en los puntos de mayor altura. Por el contrario, el método tradicional es bastante más caótico.

Analicemos ahora las direcciones de máxima y mínima curvatura. En la figura 8.21a se han dibujado las direcciones estimadas para los vectores de máxima y mínima curvatura. Puede observarse que con el método propuesto los vectores están mucho mejor alineados que con el método tradicional. En la figura 8.21b se muestra la misma representación, pero esta vez con el toroide orientado en una cierta dirección arbitraria, como el de

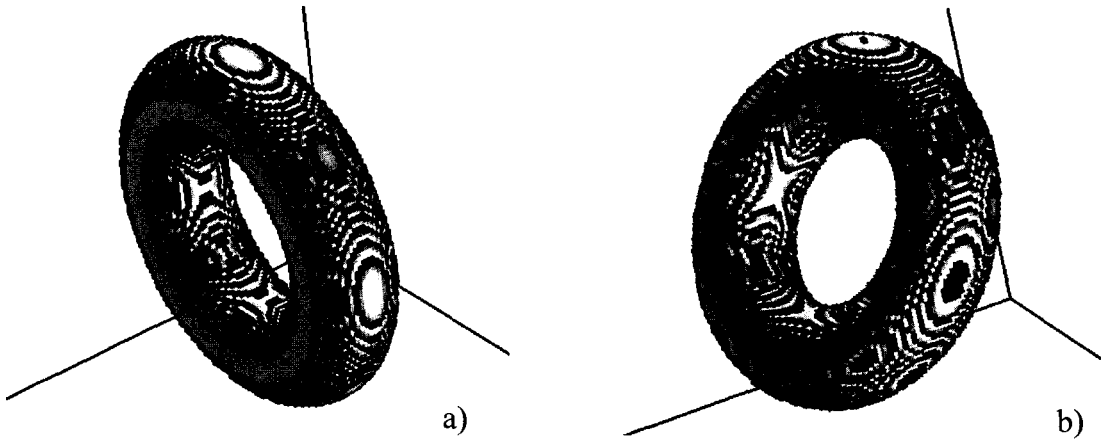


Figura 8.19: a) Toroide de radios 30 y 10 orientado en el eje z; b) Igual pero orientado en la dirección $[3, 5, 7]$

la figura 8.19b. Se aprecia cómo el método propuesto se comporta de manera casi idéntica, mientras que con el método tradicional las direcciones se vuelven mucho más desalineadas.

8.5 Conclusiones

Se ha desarrollado un esquema capaz de estimar con total precisión la orientación, curvaturas y posición subvoxel de un contorno de segundo grado en una imagen tridimensional, así como el salto de intensidad a ambos lados de la superficie. Posteriormente, a partir de dicho esquema, hemos desarrollado un método para localizar de forma general cualquier contorno sobre una imagen 3D, asumiendo que localmente dichos contornos se comportan como superficies de segundo grado, y hemos visto varios ejemplos con imágenes sintéticas conteniendo figuras geométricas conocidas.

Todas las imágenes que se han usado carecían de ruido alguno. Pero, normalmente, en casos reales, la imagen puede no venir con todos sus valores exactos. En el próximo capítulo desarrollaremos un nuevo esquema que nos permitirá lograr mejores estimaciones que el método propuesto actual cuando en la imagen exista algo de ruido.

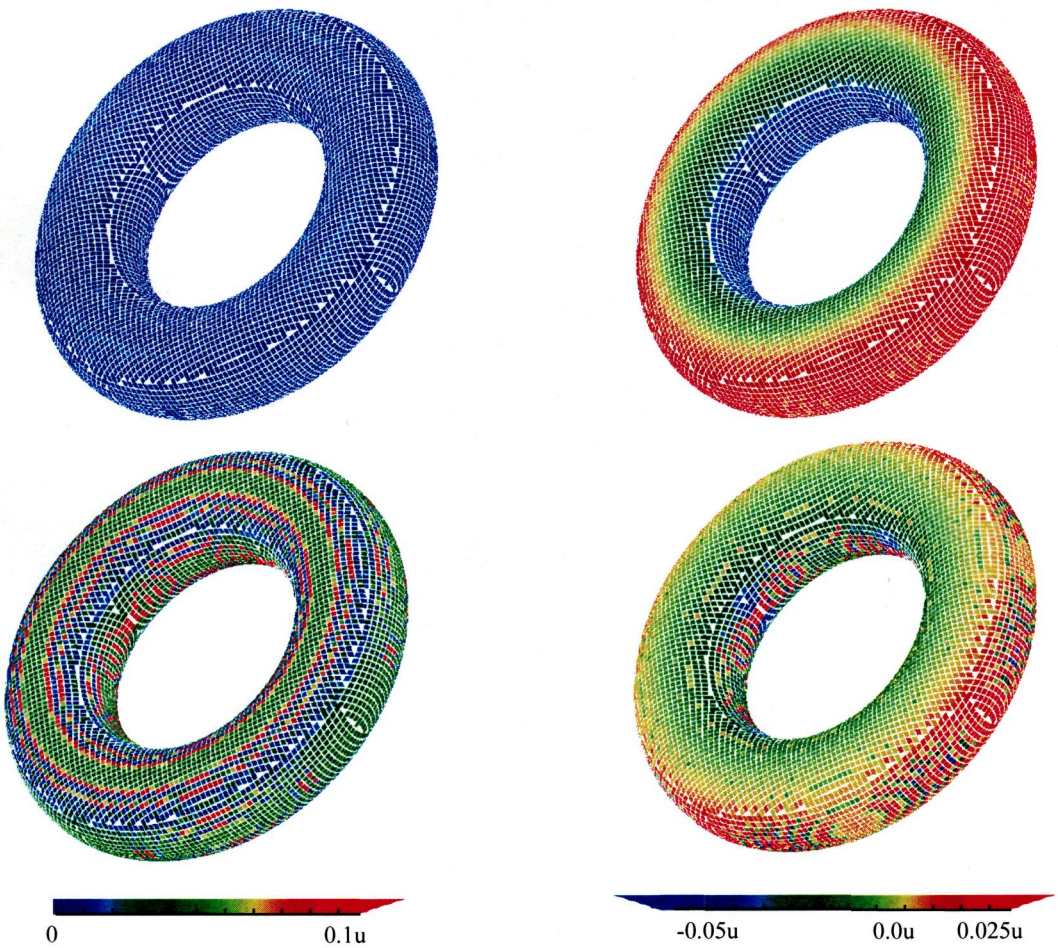


Figura 8.20: Aplicación del método propuesto (fila superior) y tradicional (fila inferior) a un toroide de radios 30 y 10. De izquierda a derecha: error cometido al estimar la curvatura máxima; valor de la curvatura mínima estimada, oscilando entre $-1/20$ y $1/40$

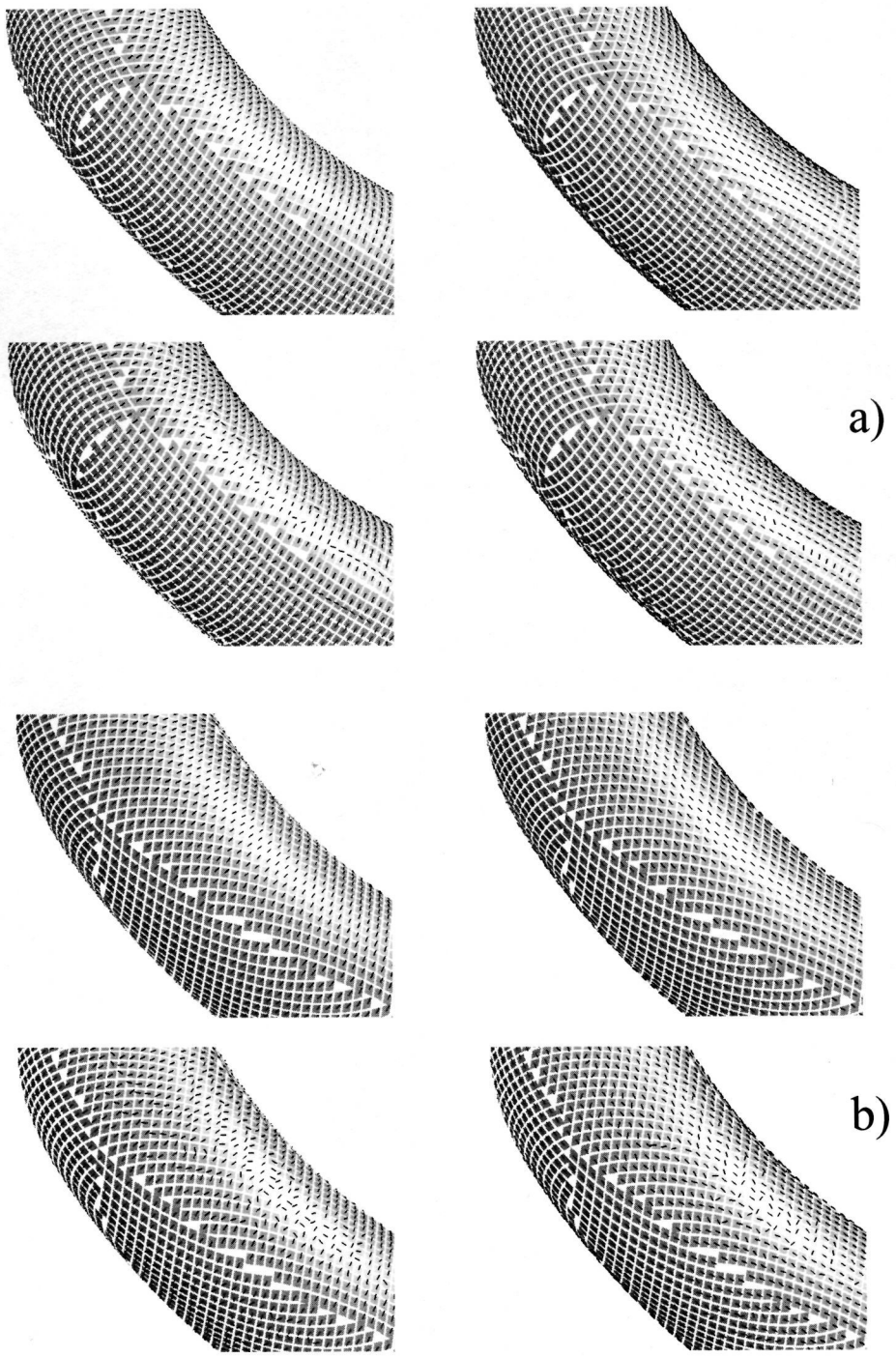


Figura 8.21: a) Direcciones de curvatura máxima (columna izquierda) y mínima (columna derecha) sobre la imagen de un toroide de radios 30 y 10 orientado en el eje z utilizando el método propuesto (fila superior) y el método tradicional (fila inferior). b) Idem con el toroide orientado en la dirección $[3, 5, 7]$

Capítulo 9

Contornos en imágenes suavizadas

- ¿Te habrás dado cuenta que para suavizar en 3D necesitarás máscaras de 27 valores?
- Sí, y por tanto seguro que en el método van a salir expresiones bastante grandes.....

OCTUBRE 2003

Los métodos que hemos desarrollado hasta ahora (de primer y segundo orden) funcionan con bastante precisión en imágenes ideales, pero producen pequeños errores cuando la imagen tiene algo de ruido, cosa que siempre va a ocurrir en mayor o menor medida en las imágenes reales debido al proceso de adquisición. Ya se comentó en el capítulo 4 que la forma clásica de eliminar ruido es convolucionar con un filtro paso bajo, a costa de perder definición en los bordes, que se vuelven más difuminados. Pero también vimos que si aplicamos nuestras técnicas a una imagen previamente suavizada, no obtendremos los valores exactos, pues la hipótesis de partida de nuestro trabajo (la intensidad de los voxels pertenecientes a los bordes de un objeto es proporcional al volumen de voxel cubierto por dicho objeto) ya no se cumple en la nueva imagen (ver sección 4.1.4).

Por tanto, llamando F a la imagen original y G a su versión suavizada, el objetivo del presente capítulo consiste en estudiar cómo es el comportamiento de una imagen con borde previamente suavizada, y así poder desarrollar un método que, aplicado a la imagen suavizada G, detecte de forma exacta el borde original presente en F.

También a efectos de comparación con otras técnicas detectoras de borde, hay que considerar que la mayor parte de ellas suavizan la imagen antes de realizar ningún cálculo, para que los resultados sean más homogéneos. Esto es debido a que, en presencia de ruido, el cálculo de parámetros locales tales como la normal al contorno o las curvaturas, puede producir resultados bastante alterados. El proceso de suavizado permite entonces que los valores de la nueva imagen sean más consistentes. Sin embargo, así como este proceso supone una buena ayuda para estas técnicas, para la nuestra ya hemos visto que supondría todo lo contrario. Por lo tanto, y para poder llevar a cabo una comparación de nuestro método con otros, es preciso adecuarlo primero para poder aplicarlo a una imagen suavizada.

La estructura de este capítulo es la siguiente: primero analizaremos las técnicas tradicionales de suavizado de imágenes 3D, para posteriormente desarrollar dos esquemas (uno con ventanas de tamaño fijo y otro variable) que nos permitan calcular la localización, orientación y curvaturas de los contornos de una imagen F, tomando como dato de entrada la imagen suavizada G. Finalmente se verán algunos ejemplos de aplicación sobre imágenes sintéticas con y sin ruido.

9.1 Suavizado tradicional de imágenes 3D

Los dispositivos de adquisición de imágenes tridimensionales introducen siempre una degradación de los valores de dicha imagen, lo que afecta de forma negativa a su calidad. Mejorar esta calidad es de vital importancia porque todas las operaciones de análisis de imágenes que se apliquen a ella lograran resultados más exactos.

Al igual que sucedía en el caso bidimensional, la forma más usual de eliminar el ruido y mejorar así la calidad de los valores consiste en convolucionar la imagen con un filtro paso bajo, o filtro de suavizado. Se han propuesto muchos filtros distintos en la literatura, tanto lineales como no lineales. Una revisión de muchos de ellos puede verse en [NIK01]. Los filtros lineales tienen la ventaja de ser más rápidos y sencillos de implementar. Sin embargo, tienen el inconveniente de difuminar los bordes de la imagen. Los filtros no lineales tratan de respetar más los bordes, pero son bastante más costosos, como puede verse en [PIT90]. Hay que tener en cuenta que todos ellos son extensiones de los filtros propuestos para el caso bidimensional, pero al añadir una tercera dimensión, el coste computacional para completar un filtrado se incrementa drásticamente.

Realmente este suavizado de la imagen no es más que una etapa de preproceso, simplemente para poder partir de una imagen de valores más regulares que la original. Esto es debido a que la mayoría de los procesos de análisis de imágenes 3D, ya complejos de por sí, requieren conocer el valor de las primeras y segundas derivadas en cada voxel, y para realizar ese cálculo acaban siempre recurriendo a simples máscaras de convolución como las vistas en el capítulo anterior para obtener derivadas parciales, aplicadas bien a la imagen original, bien a la imagen suavizada con un filtro lineal. Una discusión sobre este hecho puede verse en [LIN94].

De entre todos los filtros lineales de suavizado, al igual que ocurría en 2D, el más extendido sigue siendo un núcleo gaussiano como el visto en la sección 4.1.1, pero extrapolado para señales de 3 dimensiones.

9.1.1 Filtrado gaussiano

El núcleo gaussiano para una señal 3D viene dado por la expresión

$$h_G(x,y,z) = \frac{1}{\sigma^3\sqrt{8\pi^3}} e^{-\frac{x^2+y^2+z^2}{2\sigma^2}} \tag{9.1}$$

donde el parámetro σ controla el ancho del núcleo, y por tanto la amplitud de la difusión.

Normalmente, al trabajar con señales 3D, el filtrado suele realizarse convolucionando con una máscara, ya que el paso a frecuencia sería costosísimo. La máscara más usual es la de 3x3x3, que es la más pequeña posible (considerando máscaras centradas en el voxel). La siguiente sería de 5x5x5 y ya contabilizaría 125 voxels, demasiados para obtener el valor de un único voxel en la imagen suavizada.

Por ejemplo, para desviación igual a 1, los valores de la máscara 3x3x3 podrían ser:

$$H_G(x,y,z)_{\sigma=1} = \frac{1}{62}\left[\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}, \begin{bmatrix}2&4&2\\4&6&4\\2&4&2\end{bmatrix}, \begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}\right] \tag{9.2}$$

Otra máscara que también suele usarse es el llamado *filtro de media aritmética*, donde todos los elementos de la máscara tienen el mismo valor (1/27). Es el filtro lineal que más fuerte suaviza, pero lógicamente también es el que más difumina los bordes.

9.1.2 Detección del gradiente

Una vez la imagen está suavizada, ya se puede aplicar un esquema de cálculo del vector gradiente, como los que vimos en la sección 7.1.4, convolucionando con máscaras H_x, H_y y H_z para obtener las derivadas parciales en las direcciones x, y y z respectivamente, y luego obteniendo el valor del vector gradiente a partir de ellas. La diferencia entre estimar el gradiente directamente de la imagen original o suavizando primero la imagen radica en que los valores obtenidos en el gradiente de la imagen suavizada son más regulares que en la imagen original.

Ambos procesos (suavizado y detección del gradiente) pueden combinarse en una única convolución. Sea $f(x,y,z)$ la imagen original. Filtrar con una gaussiana $h_G(x,y,z)$ y posteriormente convolucionar con la máscara H_x para obtener la parcial en x puede expresarse como sigue

$$g(x,y,z) = f(x,y,z) * h_G(x,y,z) * H_x(x,y,z) = f(x,y,z) * H_{Gx}(x,y,z)$$

donde

$$H_{Gx}(x,y,z) = h_G(x,y,z) * H_x(x,y,z)$$

que es igual a la convolución de una máscara gaussiana con una máscara de parciales en x. Así, si usamos como máscara de la gaussiana la matriz 3x3x3 de la expresión 9.2, y como máscara H_x cualquiera de las que vimos en la sección 7.1.3, obtendríamos una única máscara 5x5x5 que, al convolucionar con la imagen original, produciría el mismo resultado que realizar los dos filtrados por separado.

9.1.3 Detectores de borde

Existen muchos detectores de borde con etapa previa de suavizado, basados en el cálculo del gradiente u otras derivadas de orden superior, entre los que cabe destacar:

Operador de Marr Propuesto en [MAR80] y extendido a 3 dimensiones en [BOM90], se define como

$$C(x,y,z) = \nabla^2 f(x,y,z) * h_G(x,y,z)$$

donde h_G es el núcleo gaussiano 3D de la expresión 9.1, y ∇^2 denota el operador laplaciano, dado por la expresión

$$\nabla^2 f(x,y,z) = \frac{\partial f(x,y,z)}{\partial x^2} + \frac{\partial f(x,y,z)}{\partial y^2} + \frac{\partial f(x,y,z)}{\partial z^2}$$

Ya que el laplaciano incorpora derivadas de segundo orden, su valor cambiará de signo en la vecindad del borde, siendo cero justo en la posición del borde. El suavizado previo de la imagen elimina el ruido de alta frecuencia que pueda producir bordes erróneos.

Detector de Canny Descrito en [CAN83], es uno de los detectores de borde más usados. En primer lugar Canny definió los criterios que debía poseer un detector de bordes: fiabilidad de la detección, exactitud en la localización y el requisito de dar una respuesta única por borde. A continuación definió una función de coste combinando estos tres criterios, y mediante cálculo variacional encontró un operador lineal óptimo con el cual convolucionar la imagen, muy similar a la primera derivada de la gaussiana, la cual en una dimensión viene dada por la expresión

$$h'_G(x) = Kxe^{-\frac{x^2}{2\sigma^2}}$$

siendo K una constante.

Para pasar a dos y a tres dimensiones se aprovecha la propiedad de separabilidad del núcleo gaussiano, pudiendo implementarse como una secuencia de tres filtros lineales unidimensionales. Por ejemplo, para calcular la componente x del vector gradiente, primero se convoluciona en la dirección x con la derivada de la gaussiana 1D, h'_G, y luego suavizamos la imagen resultante haciendo dos convoluciones 1D adicionales, una en la dirección y y otra en la z, utilizando un núcleo gaussiano normal, h_G.

Filtro de Deriche Propuesto en [DER90], consiste en un nuevo método para implementar una variación del filtro de Canny de una forma recursiva mucho más rápida. Mientras que el filtro óptimo deducido por Canny era de extensión finita, Deriche obtuvo un filtro óptimo de extensión infinita usando el mismo criterio de optimalidad seguido por Canny, el cual, una vez simplificado, podría implementarse de forma mucho más rápida, según se explica en [DER87]. El nuevo filtro tenía una forma mucho más afilada que la derivada de la gaussiana, y su expresión era del tipo

$$h_D(x) = Kxe^{-\frac{|x|}{\sigma}}$$

En [MON91] puede verse la extensión a 3D de este filtro. El punto importante de este trabajo es que el coste computacional es mucho menor que el necesitado por Canny, aún siendo de extensión infinita. De hecho, el tiempo de cálculo es independiente del valor de σ, cosa que no ocurre con el de Canny, ya que éste requiere un núcleo de convolución elevadísimo para valores altos de σ. Es por ello que el filtro de Deriche es particularmente atractivo para su uso en imágenes 3D.

9.1.4 Error cometido por el proceso de suavizado previo

En la sección 7.2 se demostró que la técnica de evaluar el gradiente por medio de las máscaras H_x, H_y y H_z no funcionaba de forma exacta, ni siquiera en imágenes ideales. De la misma manera, para el cálculo de derivadas de orden superior ocurre el mismo error. Lógicamente, si a esa imagen se le aplica un filtro gaussiano para suavizar, el resultado será más regular pero a la vez más inexacto, y más aún considerando que al aplicar una convolución discreta se comete un error con respecto a la convolución de la gaussiana en el plano continuo.

Por tanto, en la siguiente sección vamos a desarrollar un esquema que siga siendo exacto cuando la imagen ha sido suavizada.

9.2 Método suavizado de primer orden en imágenes 3D

Considerando sólamente máscaras de 3x3x3 como la de la expresión 9.2, podemos generalizar diciendo que un suavizado 3D se hace convolucionando la imagen con la máscara

$$H_G = \left[\begin{bmatrix} a_3 & a_2 & a_3 \\ a_2 & a_1 & a_2 \\ a_3 & a_2 & a_3 \end{bmatrix}, \begin{bmatrix} a_2 & a_1 & a_2 \\ a_1 & a_0 & a_1 \\ a_2 & a_1 & a_2 \end{bmatrix}, \begin{bmatrix} a_3 & a_2 & a_3 \\ a_2 & a_1 & a_2 \\ a_3 & a_2 & a_3 \end{bmatrix} \right]$$

donde se cumplen las dos condiciones siguientes:

$$\begin{aligned} a_0 &\geq a_1 \geq a_2 \geq a_3 \\ 1 &= a_0 + 6a_1 + 12a_2 + 8a_3 \end{aligned}$$

Sea F una imagen ideal como la de capítulos anteriores (ver expresión 7.5), con un contorno de primer orden que atraviesa el voxel central, de coordenadas $(0,0,0)$, que separa dos zonas de intensidad A y B, y que atraviesa el voxel central de la imagen, como la que se muestra en la figura 9.1b. Al igual que en otras ocasiones, inicialmente nos centraremos en el caso en que $|F_y| > |F_x|, |F_z|$, es decir, la componente y del vector normal en valor absoluto es mayor que las otras dos.

Dicha imagen se compone de tres zonas claramente diferenciadas: dos zonas extensas, una en la parte superior y otra en la inferior, cuyos voxels valen B y A respectivamente, y una capa central de voxels con intensidades intermedias por donde cruza la superficie del plano. Al convolucionar la imagen con la máscara de suavizado, la capa central se extiende hacia las zonas homogéneas, provocando la difuminación del borde, tal y como se ve en la figura 9.1c.

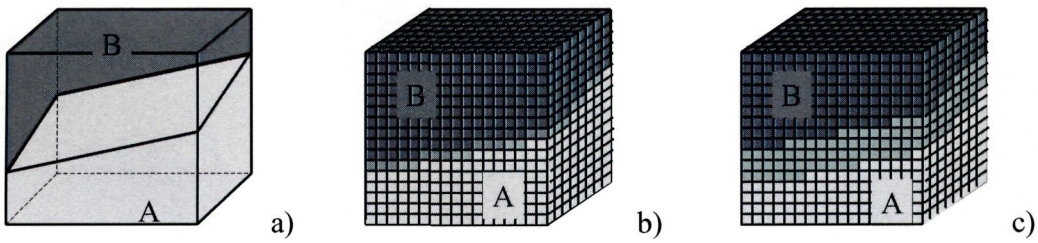


Figura 9.1: a) Contorno de primer orden. b) imagen ideal del contorno. c) imagen suavizada

Si considerásemos un caso general, es decir, todas las imágenes ideales formadas a partir de todos los planos posibles que crucen el voxel central, y donde se cumpla que $|F_y| > |F_x|, |F_z|$, obtendríamos la figura 9.2a, donde los voxels etiquetados con las intensidades A y B indican los voxels que con toda seguridad están completamente a uno u otro lado del contorno, y la zona gris clara intermedia indica todos aquellos voxels por los cuales puede pasar el contorno, y de los que por lo tanto desconocemos a priori su intensidad.

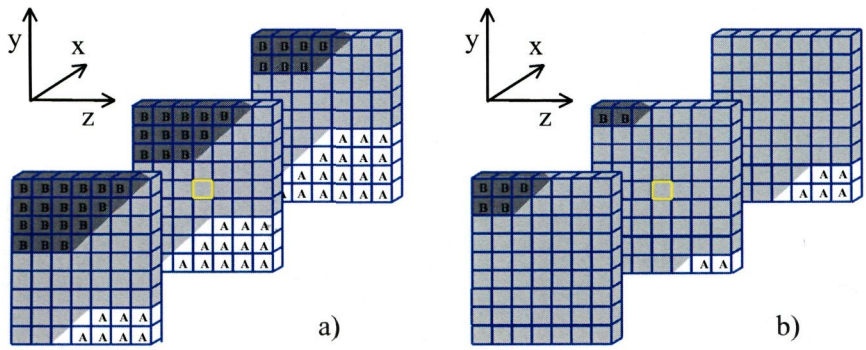


Figura 9.2: a) Imagen ideal de primer orden donde $|F_y| > |F_x|, |F_z|$ y $F_yF_x, F_yF_z \leq 0$ b) Imagen suavizada

Al convolucionar F con la máscara de suavizado H_G obtenemos la imagen G tal que

$$G = F * H_G$$

donde la intensidad de cada voxel de G viene dada por la expresión

$$\begin{aligned} G_{i,j,k} = \; & a_0 F_{i,j,k} + a_1 \left(F_{i-1,j,k} + F_{i,j-1,k} + F_{i,j+1,k} + F_{i+1,j,k} + F_{i,j,k-1} + F_{i,j,k+1}\right) + \\ & + a_2 \left(F_{i-1,j-1,k} + F_{i-1,j+1,k} + F_{i+1,j-1,k} + F_{i+1,j+1,k} + F_{i-1,j,k-1} + F_{i-1,j,k+1}\right) + \\ & + a_2 \left(F_{i+1,j,k-1} + F_{i+1,j,k+1} + F_{i,j-1,k-1} + F_{i,j+1,k-1} + F_{i,j-1,k+1} + F_{i,j+1,k+1}\right) + \\ & + a_3 \left(F_{i-1,j-1,k-1} + F_{i-1,j-1,k+1} + F_{i-1,j+1,k-1} + F_{i-1,j+1,k+1}\right) + \\ & + a_3 \left(F_{i+1,j-1,k-1} + F_{i+1,j-1,k+1} + F_{i+1,j+1,k-1} + F_{i+1,j+1,k+1}\right) \end{aligned}$$

En la figura 9.2b puede verse el aspecto que tendrá la imagen G. Como puede apreciarse, la zona de posibles valores intermedios entre A y B (color gris claro en la figura) es ahora bastante mayor que en la imagen original, debido al efecto de difuminado producido por el filtro.

El objetivo es, partiendo de la imagen G, obtener los parámetros exactos del contorno de F en el voxel central $(0, 0, 0)$. Dicho contorno puede representarse por la expresión

$$y = a + bx + cz \tag{9.3}$$

Veamos en primer lugar cómo obtener la posición exacta sobre la vertical central del voxel, a, y luego deduciremos la orientación.

9.2.1 Cálculo de la posición del contorno

Ya sabemos de capítulos anteriores que la intensidad de un voxel por donde pasa el contorno en la imagen original F viene dada por la expresión

$$F_{i,j,k} = (A - B)\frac{V_{i,j,k}}{h_x h_y h_z} + B$$

donde $V_{i,j,k}$ indica el volumen interior al voxel que se encuentra por debajo del contorno.

Llamemos $M_{i,m,n,k}$ a la suma de la columna de voxels de la imagen original F desde el voxel (i, m, k) hasta el voxel (i, n, k), siendo $m < n$. Dicha suma viene dada por la expresión

$$M_{i,m,n,k} = \sum_{j=m}^{n} F_{i,j,k} = (A - B)\frac{P_{i,m,n,k}}{h_x h_y h_z} + B\,(n - m + 1)$$

donde $P_{i,m,n,k}$ indica el volumen interior a la columna que se encuentra por debajo del contorno, y cuya expresión es

$$\begin{aligned} P_{i,m,n,k} &= \int_{(i-1/2)h_x}^{(i+1/2)h_x} \int_{(k-1/2)h_z}^{(k+1/2)h_z} \left(a + bx + cz - \frac{2m-1}{2}h_y\right) dzdx = \\ &= ah_x h_z + bih_x^2 h_z + ckh_x h_z^2 - \frac{2m-1}{2}h_x h_y h_z \end{aligned} \tag{9.4}$$

siempre y cuando se cumpla que el contorno no corte ni la base ni el techo de dicha columna.

Llamemos ahora $S_{i,m,n,k}$ a la suma de la columna de voxels de la imagen suavizada G desde el voxel (i, m, k) hasta el voxel (i, n, k), siendo $m < n$. Su expresión es

$$\begin{aligned} S_{i,m,n,k} &= \sum_{j=m}^{n} G_{i,j,k} = a_0 M_{i,m,n,k} + \\ &+a_1\left(M_{i-1,m,n,k} + M_{i,m,n,k-1} + M_{i,m-1,n+1,k} + M_{i,m+1,n-1,k} + M_{i,m,n,k+1} + M_{i+1,m,n,k}\right) + \\ &+a_2\left(M_{i-1,m,n,k-1} + M_{i-1,m-1,n+1,k} + M_{i-1,m+1,n-1,k} + M_{i-1,m,n,k+1}\right) + \\ &+a_2\left(M_{i,m-1,n+1,k-1} + M_{i,m+1,n-1,k-1} + M_{i,m-1,n+1,k+1} + M_{i,m+1,n-1,k+1}\right) + \\ &+a_2\left(M_{i+1,m,n,k-1} + M_{i+1,m-1,n+1,k} + M_{i+1,m+1,n-1,k} + M_{i+1,m,n,k+1}\right) + \\ &+a_3\left(M_{i-1,m-1,n+1,k-1} + M_{i-1,m+1,n-1,k-1} + M_{i-1,m-1,n+1,k+1} + M_{i-1,m+1,n-1,k+1}\right) + \\ &+a_3\left(M_{i+1,m-1,n+1,k-1} + M_{i+1,m+1,n-1,k-1} + M_{i+1,m-1,n+1,k+1} + M_{i+1,m+1,n-1,t+1}\right) \end{aligned}$$

Si realizamos la suma de todos los voxels de la columna central de la imagen suavizada G que a priori pueden tener valores intermedios entre A y B (11 en total), obtenemos que

$$S_M = S_{0,-5,5,0} = \frac{a}{h_y}(A - B) + \frac{11}{2}(A + B)$$

De esta expresión ya podemos despejar el valor de a, el cual resulta ser

$$a = \frac{h_y}{2(A-B)}\left(2S_M - 11(A+B)\right)$$

9.2.2 Cálculo de la orientación

Para deducir la orientación del plano haremos uso de las columnas laterales, de forma similar a como se hacía en el caso 2D. Tomemos primero la diferencia de las columnas laterales en la dirección x. Atendiendo a la figura 9.2, que muestra el caso donde las derivadas parciales de la imagen suavizada, G_x y G_y, tienen distinto signo, podríamos tomar la columna derecha desplazada un voxel hacia arriba, y la columna izquierda un voxel hacia abajo. Para el caso contrario, cuando $G_xG_y > 0$, haríamos lo contrario. Por lo tanto, llamando α al signo del producto G_xG_y, podemos evaluar la expresión

$$S_R - S_L = S_{1,-5-\alpha,5-\alpha,0} - S_{-1,-5+\alpha,5+\alpha,0} = \frac{2}{h_y}(bh_x + \alpha h_y)(A-B)$$

De esta expresión deducimos el valor de b

$$b = \frac{h_y}{h_x}\left(\frac{S_R - S_L}{2(A-B)} - \alpha\right)$$

Para el valor de c hacemos lo mismo, pero tomando el otro par de columnas laterales. Así, llamando β al signo del producto G_yG_z, evaluamos la expresión

$$S_F - S_B = S_{0,-5-\beta,5-\beta,1} - S_{0,-5+\beta,5+\beta,-1} = \frac{2}{h_y}(ch_z + \beta h_y)(A-B)$$

De aquí deducimos c

$$c = \frac{h_y}{h_z}\left(\frac{S_F - S_B}{2(A-B)} - \beta\right)$$

9.2.3 Cálculo del salto de intensidad

Para evaluar las expresiones anteriores de los coeficientes del plano, necesitamos una estimación de los valores de intensidad A y B. Para ello podemos utilizar los voxels de la vecindad que sabemos con toda seguridad que tendrán uno de estos dos valores. Atendiendo a la figura 9.2b, los voxels a utilizar serían los mostrados en la figura 9.3, cuyas expresiones son

$$\begin{aligned} A &\approx \frac{1}{4}\left(G_{-\alpha,-5,0} + G_{-\alpha,-5,-\beta} + G_{0,-5,-\beta} + G_{-\alpha,-4,-\beta}\right) \\ B &\approx \frac{1}{4}\left(G_{\alpha,5,0} + G_{\alpha,5,\beta} + G_{0,5,\beta} + G_{\alpha,4,\beta}\right) \end{aligned} \tag{9.5}$$

donde α y β son los mismos que se utilizaron para las expresiones de b y c respectivamente.

9.2.4 Lema final

Los resultados de los apartados anteriores se resumen en el siguiente lema:

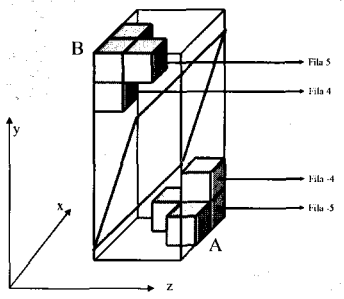


Figura 9.3: Voxels utilizados para estimar las intensidades A y B para el caso $G_xG_y, G_yG_z < 0$

Lema 9.1 *Sea F una imagen 3D por cuyo voxel $(0,0,0)$ pasa un contorno plano que separa dos zonas de intensidad A (debajo del contorno) y B (encima del contorno), y que cumple que $|F_y|\,h_y > |F_x|\,h_x, |F_z|\,h_z$. Dicha imagen ha sido suavizada con una máscara de suavizado 3x3x3, dando como resultado la imagen G. A partir de los datos de la imagen suavizada G es posible obtener los parámetros del contorno de la imagen original F (salto de intensidad, posición sub-voxel y vector normal), realizando los siguientes pasos:*

a) obtenemos las sumas de las columnas de voxels de la vecindad, de la siguiente manera:

$$\begin{array}{ll} S_M = \sum_{j=-5}^{5} F_{0,j,0} & \\ S_R = \sum_{j=-5-\alpha}^{5-\alpha} F_{1,j,0} & S_B = \sum_{j=-5+\beta}^{5+\beta} F_{0,j,-1} \\ S_F = \sum_{j=-5-\beta}^{5-\beta} F_{0,j,1} & S_L = \sum_{j=-5+\alpha}^{5+\alpha} F_{-1,j,0} \end{array}$$

donde

$$\alpha = \begin{cases} 1 & \text{si } G_xG_y \geq 0 \\ -1 & \text{si } G_xG_y < 0 \end{cases} \qquad \beta = \begin{cases} 1 & \text{si } G_zG_y \geq 0 \\ -1 & \text{si } G_zG_y < 0 \end{cases}$$

b) obtenemos también una estimación para los valores de intensidad a ambos lados del borde

$$\begin{aligned} A &= \frac{1}{4}\left(G_{-\alpha,-5,0} + G_{-\alpha,-5,-\beta} + G_{0,-5,-\beta} + G_{-\alpha,-4,-\beta}\right) \\ B &= \frac{1}{4}\left(G_{\alpha,5,0} + G_{\alpha,5,\beta} + G_{0,5,\beta} + G_{\alpha,4,\beta}\right) \end{aligned}$$

c) a continuación detectamos los coeficientes del plano $y = a + bx + cz$ como sigue:

$$\begin{aligned} a &= \tfrac{h_y}{2(A-B)}\left(2S_M - 11\,(A+B)\right) \\ b &= \tfrac{h_y}{h_x}\left(\tfrac{S_R - S_L}{2(A-B)} - \alpha\right) \\ c &= \tfrac{h_y}{h_z}\left(\tfrac{S_F - S_B}{2(A-B)} - \beta\right) \end{aligned}$$

d) la posición y el vector normal se calculan a partir de la expresión del plano

$$\begin{aligned} \text{corte con la vertical central del voxel} &: (0, a, 0) \\ \text{vector normal} &: N = \frac{A-B}{\sqrt{1+b^2+c^2}}\begin{bmatrix} b & -1 & c \end{bmatrix} \end{aligned}$$

Con este lema queda demostrado que, usando exclusivamente la información de la imagen suavizada G, independientemente de los valores usados en la máscara de suavizado, pueden obtenerse de forma exacta los parámetros del contorno. El inconveniente con respecto al método original de la sección 7.8.4 es que, debido al efecto de difuminado provocado por la convolución, hay que usar vecinos más alejados. Sin embargo, al final de este capítulo veremos una modificación para no tener que llegar tan lejos.

Como el método es válido para cualquier máscara de suavizado, esto incluye a la máscara

$$H_G = \left[\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \right]$$

Esta máscara produce de nuevo la imagen original, con lo cual podemos afirmar que este método seguiría funcionando si lo aplicamos directamente en la imagen original F. Sin embargo, hay que decir que en los casos en que F no tenga ruido, es más conveniente aplicar el método desarrollado en capítulos anteriores, pues la información que se usa para estimar el contorno en cada pixel es más local (una vecindad más reducida).

9.3 Método suavizado de segundo orden en imágenes 3D

Para poder estimar la curvatura del contorno, simplemente cambiaremos la expresión lineal que acabamos de considerar para representar dicho contorno (ecuación 9.3) por una expresión cuadrática, del tipo

$$y = a + bx + cz + dx^2 + fxz + gz^2$$

de la misma manera que hicimos en el capítulo previo. Por tanto, sea ahora F una imagen ideal como la de capítulos anteriores, con un contorno de segundo orden, tal que $|F_y| > |F_x|, |F_z|$, es decir, la componente y del vector normal en valor absoluto es mayor que las otras dos. Dicho contorno cruza el voxel central, de coordenadas $(0,0,0)$ y separa dos zonas de intensidad A y B, como muestra la figura 9.4.

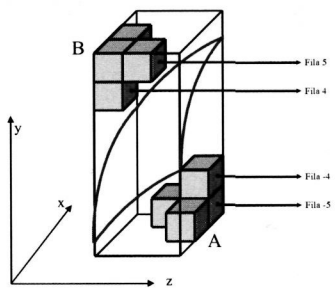


Figura 9.4: Contorno de segundo orden

En principio, vamos a considerar que los voxels por los cuales puede pasar el contorno, cuyas intensidades son a priori desconocidas, son los mismos que para el caso lineal, tal y como se vio en la figura 9.2. Esto significa que para contornos con curvatura muy grande podríamos estar cometiendo un error en esta consideración, aunque en la gran mayoría de los casos este hecho no va a ocurrir.

9.3.1 Planteamiento del sistema de ecuaciones

Si rehacemos todo el desarrollo previo de la sección anterior, todas las expresiones se mantienen igual excepto la de $P_{i,m,n,k}$ (ecuación 9.4), que indicaba el volumen interior a la columna que se encuentra por debajo del

contorno. Ahora su expresión será:

$$\begin{aligned} P_{i,m,n,k} &= \int_{(i-1/2)h_x}^{(i+1/2)h_x} \int_{(k-1/2)h_z}^{(k+1/2)h_z} \left(a + bx + cz + dx^2 + fxz + gz^2 - \frac{2m-1}{2} h_y\right) dzdx = \\ &= ah_x h_z + bih_x^2 h_z + ckh_x h_z^2 + dh_x^3 h_z \left(i^2 + \frac{1}{12}\right) + fikh_x^2 h_z^2 + gh_x h_z^3 \left(k^2 + \frac{1}{12}\right) - \frac{2m-1}{2} h_x h_y h_z \end{aligned} \tag{9.6}$$

Como son seis los coeficientes que deseamos averiguar, son seis las ecuaciones que debemos plantear, y son éstas:

$$\begin{array}{lll} S_M = S_{0,-5,5,0}; & S_L = S_{-1,-5+\alpha,5+\alpha,0}; & S_R = S_{1,-5-\alpha,5-\alpha,0} \\ S_B = S_{0,-5+\beta,5+\beta,-1}; & S_F = S_{0,-5-\beta,5-\beta,1}; & S_D = S_{-1,-5,5,\gamma} + S_{1,-5,5,-\gamma} \end{array} \tag{9.7}$$

donde

$$\alpha = \begin{cases} 1 & \text{si } G_x G_y \geq 0 \\ -1 & \text{si } G_x G_y < 0 \end{cases} \qquad \beta = \begin{cases} 1 & \text{si } G_z G_y \geq 0 \\ -1 & \text{si } G_z G_y < 0 \end{cases} \qquad \gamma = \begin{cases} 1 & \text{si } G_x G_z \geq 0 \\ -1 & \text{si } G_x G_z < 0 \end{cases}$$

El parámetro γ toma este valor para evitar tomar en cuenta las columnas en donde mayor probabilidad hay de que el contorno toque la base o el techo de la ventana, de igual forma que se hizo en la sección 8.2.4.

Con esto obtenemos un sistema lineal de seis ecuaciones y seis incógnitas, cuya solución es:

$$\begin{aligned} a &= \frac{h_y}{2(A-B)} (2S_M - 11(A+B)) + \frac{(1 + 24a_{001} + 96a_{011} + 96a_{111}) h_y}{24(A-B)} (4S_M - S_B - S_F - S_L - S_R) \\ b &= \frac{h_y}{h_x} \left(\frac{S_R - S_L}{2(A-B)} - \alpha\right) \\ c &= \frac{h_y}{h_z} \left(\frac{S_F - S_B}{2(A-B)} - \beta\right) \\ d &= \frac{h_y}{2(A-B)h_x^2} (S_R + S_L - 2S_M) \\ f &= \frac{\gamma h_y}{2h_x h_z (A-B)} (S_F + S_L + S_B + S_R - S_D - 2S_M) \\ g &= \frac{h_y}{2(A-B)h_z^2} (S_F + S_B - 2S_M) \end{aligned}$$

Podemos apreciar una gran diferencia con respecto al método lineal de la sección anterior, y es que para poder obtener el valor de a es preciso conocer cuáles fueron los valores empleados en la máscara de suavizado, cosa que no ocurría antes, ya que era independiente de los valores de la máscara.

Por otro lado, los coeficientes de grado 1, b y c, se obtienen con idéntica expresión que en el método lineal. Esto significa que el vector normal obtenido es exactamente el mismo en ambos métodos. Sin embargo, el desplazamiento vertical del contorno es parecido pero con un cierto factor de corrección, en función de la curvatura. De hecho, es fácil ver que la expresión obtenida para a puede ponerse también como

$$a = \frac{h_y}{2(A-B)} (2S_M - 11(A+B)) - k\left(h_x^2 d + h_z^2 g\right)$$

donde k es una constante que depende de los pesos de la máscara de suavizado, dada por la expresión

$$k = \frac{1 + 24a_1 + 96a_2 + 96a_3}{12} \tag{9.8}$$

De esta manera, podemos ver como el método lineal es un caso particular del método de segundo orden, forzando a que $d = f = g = 0$.

9.3.2 Lema final

Las expresiones anteriores se resumen en el siguiente lema:

Lema 9.2 *Sea F una imagen 3D por cuyo voxel $(0,0,0)$ pasa un contorno que separa dos zonas de intensidad A (debajo del contorno) y B (encima del contorno), y que cumple que $|F_y|\,h_y > |F_x|\,h_x, |F_z|\,h_z$. Dicha imagen ha sido suavizada con una máscara de suavizado H_G,* tal que

$$H_G = \left[\begin{bmatrix} a_3 & a_2 & a_3 \\ a_2 & a_1 & a_2 \\ a_3 & a_2 & a_3 \end{bmatrix}, \begin{bmatrix} a_2 & a_1 & a_2 \\ a_1 & a_0 & a_1 \\ a_2 & a_1 & a_2 \end{bmatrix}, \begin{bmatrix} a_3 & a_2 & a_3 \\ a_2 & a_1 & a_2 \\ a_3 & a_2 & a_3 \end{bmatrix} \right]$$

dando como resultado la imagen G. A partir de los datos de la imagen suavizada G es posible obtener los parámetros del contorno de la imagen original F (salto de intensidad, posición sub-voxel, vector normal y curvaturas), realizando los siguientes pasos:

a) obtenemos las sumas de las columnas de voxels de la vecindad, de la siguiente manera:

$$\begin{array}{ccc} S_M = S_{0,-5,5,0}; & S_L = S_{-1,-5+\alpha,5+\alpha,0}; & S_R = S_{1,-5-\alpha,5-\alpha,0} \\ S_B = S_{0,-5+\beta,5+\beta,-1}; & S_F = S_{0,-5-\beta,5-\beta,1}; & S_D = S_{-1,-5,5,\gamma} + S_{1,-5,5,-\gamma} \end{array}$$

donde

$$\alpha = \begin{cases} 1 & \text{si } G_xG_y \geq 0 \\ -1 & \text{si } G_xG_y < 0 \end{cases} \qquad \beta = \begin{cases} 1 & \text{si } G_zG_y \geq 0 \\ -1 & \text{si } G_zG_y < 0 \end{cases} \qquad \gamma = \begin{cases} 1 & \text{si } G_xG_z \geq 0 \\ -1 & \text{si } G_xG_z < 0 \end{cases}$$

b) obtenemos también una estimación para los valores de intensidad a ambos lados del borde

$$\begin{aligned} A &= \frac{1}{4}\left(G_{-\alpha,-5,0} + G_{-\alpha,-5,-\beta} + G_{0,-5,-\beta} + G_{-\alpha,-4,-\beta}\right) \\ B &= \frac{1}{4}\left(G_{\alpha,5,0} + G_{\alpha,5,\beta} + G_{0,5,\beta} + G_{\alpha,4,\beta}\right) \end{aligned}$$

c) a continuación detectamos los coeficientes del paraboloide $y = a + bx + cz + dx^2 + fxz + gz^2$ como sigue:

$$\begin{aligned} b &= \frac{h_y}{h_x}\left(\frac{S_R - S_L}{2\,(A-B)} - \alpha\right) \\ c &= \frac{h_y}{h_z}\left(\frac{S_F - S_B}{2\,(A-B)} - \beta\right) \\ d &= \frac{h_y}{2\,(A-B)\,h_x^2}\left(S_R + S_L - 2S_M\right) \\ f &= \frac{\gamma h_y}{2h_xh_z\,(A-B)}\left(S_F + S_L + S_B + S_R - S_D - 2S_M\right) \\ g &= \frac{h_y}{2\,(A-B)\,h_z^2}\left(S_F + S_B - 2S_M\right) \\ a &= \frac{h_y}{2\,(A-B)}\left(2S_M - 11\,(A+B)\right) - k\left(h_x^2 d + h_z^2 g\right) \end{aligned}$$

donde

$$k = \frac{1 + 24a_1 + 96a_2 + 96a_3}{12}$$

d) la posición y el vector normal se calculan a partir de la expresión del paraboloide

$$\begin{aligned} \text{corte con la vertical central del voxel} &: (0, a, 0) \\ \text{vector normal} &: N = \frac{A - B}{\sqrt{1 + b^2 + c^2}} \begin{bmatrix} b & -1 & c \end{bmatrix} \end{aligned}$$

e) las curvaturas también se calculan a partir de la expresión del paraboloide, aplicando el lema 8.1

Hay que tener en cuenta que el anterior lema sólo es aplicable para el caso $|G_y|\, h_y > |G_x|\, h_x, |G_z|\, h_z$. Cuando esto no ocurra, el esquema será idéntico, pero intercambiando las variables entre sí. De esta manera, si la parcial mayor fuera, por ejemplo, la de la dirección x, es decir, se cumpliese que $|G_z|\, h_x > |G_y|\, h_y, |G_z|\, h_z$, el lema a aplicar sería equivalente pero realizando las sumas por filas en lugar de por columnas, e intercambiando los índices y las variables x e y, así como las componentes de los vectores.

9.3.3 Casos que producen error

Hay que tener en cuenta que contornos con curvatura muy alta pueden provocar error. Esto es debido a que no se cumple la suposición inicial que se ha hecho para el cálculo de la expresión de los volúmenes $P_{i,m,n,k}$ (ecuacion 9.6). Recordemos que para obtener esta expresión supusimos que el contorno nunca corta la base ni el techo de la columna. Si esto no es así, el método no sería exacto.

En la figura 9.5a se muestra la situación correcta que debe ocurrir en el interior de una columna para que la expresión 9.6 indique un volumen correcto. Si el contorno tocase la base o el techo de la columna, como se ve en la figura 9.5b, el valor sería incorrecto.

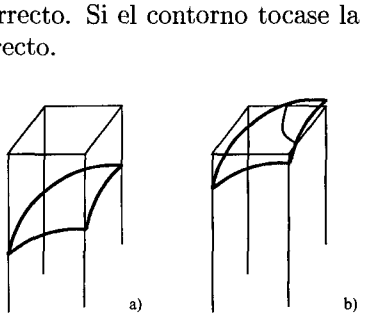


Figura 9.5: El cálculo del volumen interior a la columna que pasa por debajo del contorno según la expresión 9.6 es correcto en la figura a) pero incorrecto en la b)

Sin embargo, a efectos numéricos, esto ocurrirá en las columnas más alejadas de la columna central, que son las que menos peso tienen debido a los valores de la máscara de suavizado. Por tanto, el error será bastante pequeño. En realidad, éste está relacionado con el volumen bajo el contorno que queda fuera de la columna, escalado por el coeficiente de la máscara de suavizado que le corresponde al voxel donde ocurre la intersección. A menor sea este volumen, menor será el error.

9.3.4 Algoritmo propuesto

Al igual que en los otros métodos propuestos, en primer lugar habrá que detectar qué voxels son etiquetados como borde. Para ello, obtendremos las tres imágenes de parciales de la imagen suavizada, G_x, G_y y G_z, y nos centraremos en aquellos voxels (i,j,k) donde

$$G_x^2(i,j,k) + G_y^2(i,j,k) + G_z^2(i,j,k) > \delta^2$$

y donde además la derivada parcial de mayor valor absoluto sea máxima en su respectiva fila o columna. Una vez tenemos los voxels identificados, aplicaremos alrededor de cada uno el lema 9.2 para obtener los parámetros del contorno (posición, vector normal, salto de intensidad y curvaturas principales).

El pseudo-código de nuestro algoritmo podría ser el siguiente:

Funcion ContornosSuavizados3D(delta)

```
Calcular Gx, Gy, Gz
Para todos los voxels (i,j,k) de la imagen
   Si Gx(i,j,k)*Gx(i,j,k)+Gy(i,j,k)*Gy(i,j,k)+Gz(i,j,k)*Gz(i,j,k)<delta*delta Continuar
   Si (|Gx(i,j,k)|hx>|Gy(i,j,k)|hy) y (|Gx(i,j,k)|hx>|Gz(i,j,k)|hz) Entonces
      Si |Gx(i,j,k)| no es maximo en su fila Continuar
      Aplicar lema 9.2 con la variable x como direccion principal
   Fin Si
   Si (|Gy(i,j,k)|hy>|Gx(i,j,k)|hx) y (|Gy(i,j,k)|hy>|Gz(i,j,k)|hz) Entonces
      Si |Gy(i,j,k)| no es maximo en su columna Continuar
      Aplicar lema 9.2 con la variable y como direccion principal
   Fin Si
   Si (|Gz(i,j,k)|hz>|Gx(i,j,k)|hx) y (|Gz(i,j,k)|hz>|Gy(i,j,k)|hy) Entonces
      Si |Gz(i,j,k)| no es maximo en su fila Continuar
      Aplicar lema 9.2 con la variable z como direccion principal
   Fin Si
Fin Para
```

9.3.5 Ejemplos

Vamos a aplicar el método a algunas de las superficies geométricas sintéticas del capítulo anterior, a las que previamente añadiremos ruido, y compararemos los resultados de nuevo con la técnica tradicional propuesta por Álvarez (ver sección 8.1.1), pero aplicada sobre la imagen suavizada.

Esferas

Comencemos por la imagen de una esfera de radio 20 voxels, la cual hemos suavizado con una máscara 3x3x3. En la tabla 9.1 se muestra en las tres primeras filas el error cometido en la estimación de la posición sub-voxel, el salto de intensidad, la dirección del vector normal y las curvaturas principales en cada voxel borde usando tres métodos diferentes: el método de Álvarez, el método cuadrático propuesto en el capítulo anterior (MP), y el método propuesto del presente capítulo (MPIS).

Vemos efectivamente que el método cuadrático previo comete error en todas las magnitudes cuando la imagen ha sido suavizada, ya que la hipótesis de partida ya no es cierta en los bordes. Sin embargo, el método suavizado logra una gran precisión, incluso en los valores de curvatura. Por otro lado, el método tradicional comete un error altísimo, siendo el más grave el del salto de intensidad a ambos lados del borde (70 unidades de error medio).

		Error pos.		Error salto		Error dir.		Curv. mín.		Curv. máx.	
Imagen	**Método**	Med.	Máx.	Med.	Máx.	Med.	Máx.	Med.	Rad.	Med.	Rad.
sin ruido	Tradic.	–	–	71.7	80.6	1.10	1.80	0.025	40.6	0.028	35.4
	MP	3.81	10.5	0.95	6.42	0.31	0.96	0.049	20.3	0.051	19.7
	MPIS	0.11	0.65	0.00	0.00	0.24	0.92	0.050	20.0	0.051	19.7
ruido 5	Tradic.	–	–	71.7	82.7	1.15	2.57	0.023	42.8	0.030	33.9
	MP	3.87	12.4	1.28	8.21	0.64	2.28	0.042	23.6	0.058	17.1
	MPIS	1.63	8.19	1.07	4.23	0.63	2.15	0.043	23.2	0.057	17.6
ruido 10	Tradic.	–	–	71.9	86.8	1.26	3.68	0.021	46.9	0.032	31.7
	MP	4.02	17.4	1.92	9.79	1.11	3.86	0.037	27.3	0.064	15.7
	MPIS	3.12	16.8	1.94	7.70	1.15	4.33	0.035	28.4	0.066	15.2

Tabla 9.1: Errores cometidos por cada método al obtener el contorno de una esfera de radio 20 con diversas magnitudes de ruido. MP: método propuesto original. MPIS: método propuesto para imágenes suavizadas.

En las siguientes tres filas de la tabla se le ha añadido un ruido de magnitud 5 a la imagen, antes de aplicar el suavizado. Los errores son mayores en todas las magnitudes y para los tres métodos, aunque el método suavizado propuesto sigue siendo el que mejor resultados obtiene.

Finalmente para las tres últimas filas de la tabla se le ha añadido a la imagen un ruido de magnitud 10. Los errores ahora son aún mayores, y ya hay menos diferencia entre los tres métodos, aunque el salto de intensidad sigue siendo bastante más preciso con los métodos propuestos.

En la figura 9.6 se muestra el error cometido en la estimación de la dirección del vector normal en cada voxel para cada una de las imágenes (sin ruido en la columna izquierda, ruido 5 en la central y 10 en la derecha). La fila superior muestra el error cometido cuando se utiliza el método cuadrático previo sobre la imagen original con ruido sin suavizar. Vemos que el error es bastante grande cuando hay ruido, y esa es la razón por la que en imágenes reales debe suavizarse siempre antes de realizar ninguna estimación.

En el resto de filas se utiliza como entrada la imagen después de suavizar: la fila central muestra el error con el método tradicional, y la fila inferior con el método suavizado propuesto. Puede verse cómo cuando el ruido no es muy alto, el error cometido es menor con nuestro método.

Otro detalle a mencionar es el pequeño error producido, incluso sin existir ruido, en las zonas de la superficie de la esfera más alejada de los ejes centrales. Como se comentó en el capítulo anterior, esto es debido a la mayor diferencia existente en esas zonas entre un trozo de esfera y un trozo de paraboloide. Este efecto además es más pronunciado cuando la imagen se ha suavizado, ya que aumenta el tamaño de la vecindad escogida para estimar la superficie en un voxel. Solventaremos este inconveniente en el próximo capítulo.

Cilindros

Hagamos lo mismo con un cilindro de radio 15 orientado en un eje arbitrario $u = [3, 5, 7]$, y apliquemos los tres métodos a la imagen suavizada a la que se le ha añadido un ruido previo de una cierta magnitud (0, 5 y 10). En la tabla 9.2 se muestran los resultados. De nuevo puede apreciarse que cuando el ruido es nulo o muy pequeño, el método suavizado propuesto es más preciso que los otros dos.

Toroides

Por último probemos con la imagen de un toroide de radio exterior 30 y radio interior 10. La curvatura mínima en esta figura oscila entre -1/20 en los puntos interiores hasta 1/40 en los exteriores. En la figura 9.7 se muestra el valor de la curvatura mínima sobre la superficie del toroide, para distintas magnitudes de ruido (0, 5 y 10 de

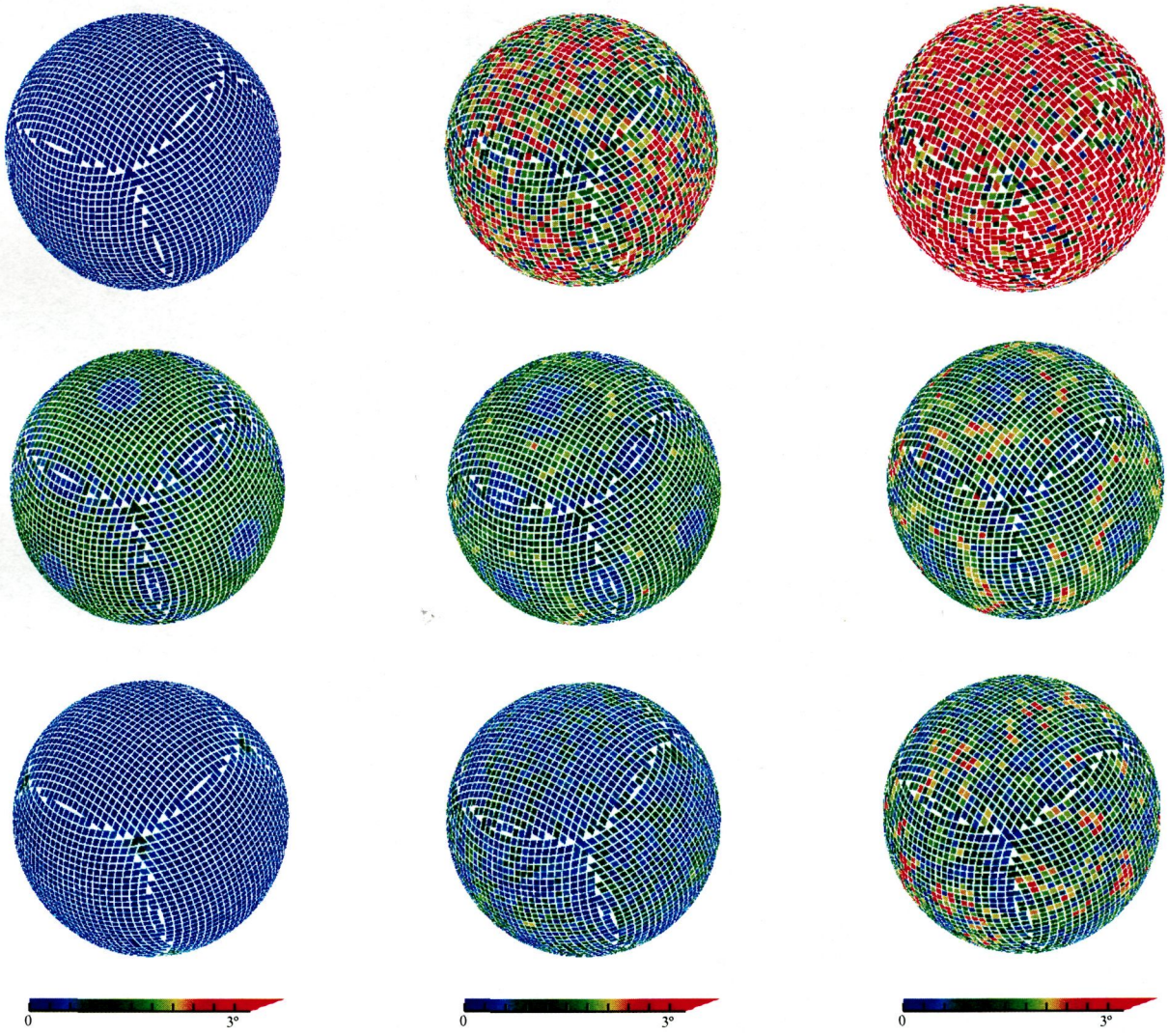


Figura 9.6: Error cometido la dirección del vector normal sobre una esfera de radio 20, a la que se ha añadido ruido de magnitud (de izquierda a derecha) 0, 5 y 10. De arriba a abajo: método cuadrático previo aplicado a la imagen original sin suavizar, método tradicional aplicado a la imagen suavizada, y método suavizado propuesto aplicado a la imagen suavizada

		Curvatura mínima			Curvatura máxima		
Imagen	**Método**	Media	Radio	Error dir.	Media	Radio	Error dir.
sin ruido	Tradic.	0.000	∞	2.06	0.035	28.4	1.70
	MP	0.000	∞	0.41	0.067	14.9	0.39
	MPIS	0.000	∞	0.12	0.067	14.8	0.11
ruido 5	Tradic.	0.001	1446	4.18	0.035	28.3	4.12
	MP	0.000	∞	3.68	0.067	14.8	3.63
	MPIS	0.000	∞	3.32	0.068	14.7	3.06
ruido 10	Tradic.	0.002	425	8.50	0.036	28.1	8.39
	MP	0.002	588	7.14	0.068	14.6	7.04
	MPIS	0.000	∞	5.72	0.069	14.4	5.51

Tabla 9.2: Errores cometidos por cada método al obtener el contorno de un cilindro de radio 15 orientado en la dirección [3,5,7] con diversas magnitudes de ruido. MP: método propuesto original. MPIS: método propuesto para imágenes suavizadas.

izquierda a derecha). En la fila superior se muestra la estimación sobre la imagen original sin suavizar con el método cuadrático previo, donde se observa el caos en la estimación en presencia de ruido.

En las filas central e inferior se muestran los resultados de aplicar el método tradicional y el suavizado propuesto respectivamente. Puede apreciarse como la distribución de los valores de curvatura es más precisa en este último.

9.4 Método de límites variables

El principal inconveniente del método suavizado propuesto es que usa una vecindad demasiado grande para estimar los parámetros del borde en cada voxel. El problema radica en que, según la hipótesis de partida, no debe existir ningún otro contorno cercano dentro de esa vecindad. En las imágenes sintéticas de ejemplo que hemos usado hasta ahora no ocurría ese problema, porque no existía ninguna zona en donde hubiese contornos cercanos entre sí. Sin embargo, en una imagen real, cualquier situación puede ocurrir.

En la fila superior de la figura 9.8 se muestra la situación normal cuando sólo existe un único contorno en la vecindad[1], y cómo son los valores de los voxels en la imagen original y en la suavizada. Para cada voxel borde, el detector trabaja con una cierta vecindad en la imagen suavizada para estimar los parámetros del contorno, y para ello parte de que las zonas superior e inferior de la ventana (para un caso vertical) se corresponden con los valores de intensidad a cada lado del contorno.

Pero ¿qué ocurre cuando existe un segundo contorno cercano al que estamos tratando de estimar? En el ejemplo de la fila inferior de la misma figura se aprecia cómo la presencia de este segundo contorno produce que, tras el proceso de suavizado, los valores de la zona superior de la ventana no coincidan con la intensidad B. Esta variación hará que al aplicar el método detector, los valores estimados para el contorno sean erróneos.

En realidad, el problema viene de usar una vecindad tan larga para estimar el contorno. Atendiendo a las expresiones para las sumas del lema 9.2 vemos que llegan a usarse hasta 6 voxels alejados en vertical del voxel central (ecuación 9.7). Esto era así para garantizar en todos los casos que el contorno no tocase nunca los extremos superior e inferior de cada columna de voxels de la ventana. A su vez, para estimar los valores de intensidad A y B según el mismo lema también se usaban voxels bastante alejados (ecuación 9.5). En la figura 9.9 se muestra cómo este lema permite obtener con exactitud los parámetros del contorno cuando el contorno está aislado (figura a), pero comete un error cuando hay otro contorno cercano (figura b) debido a una mala estimación de uno de los valores de intensidad.

[1] Realmente la gráfica representa un caso 2D, aunque explica igualmente la situación en imágenes 3D.

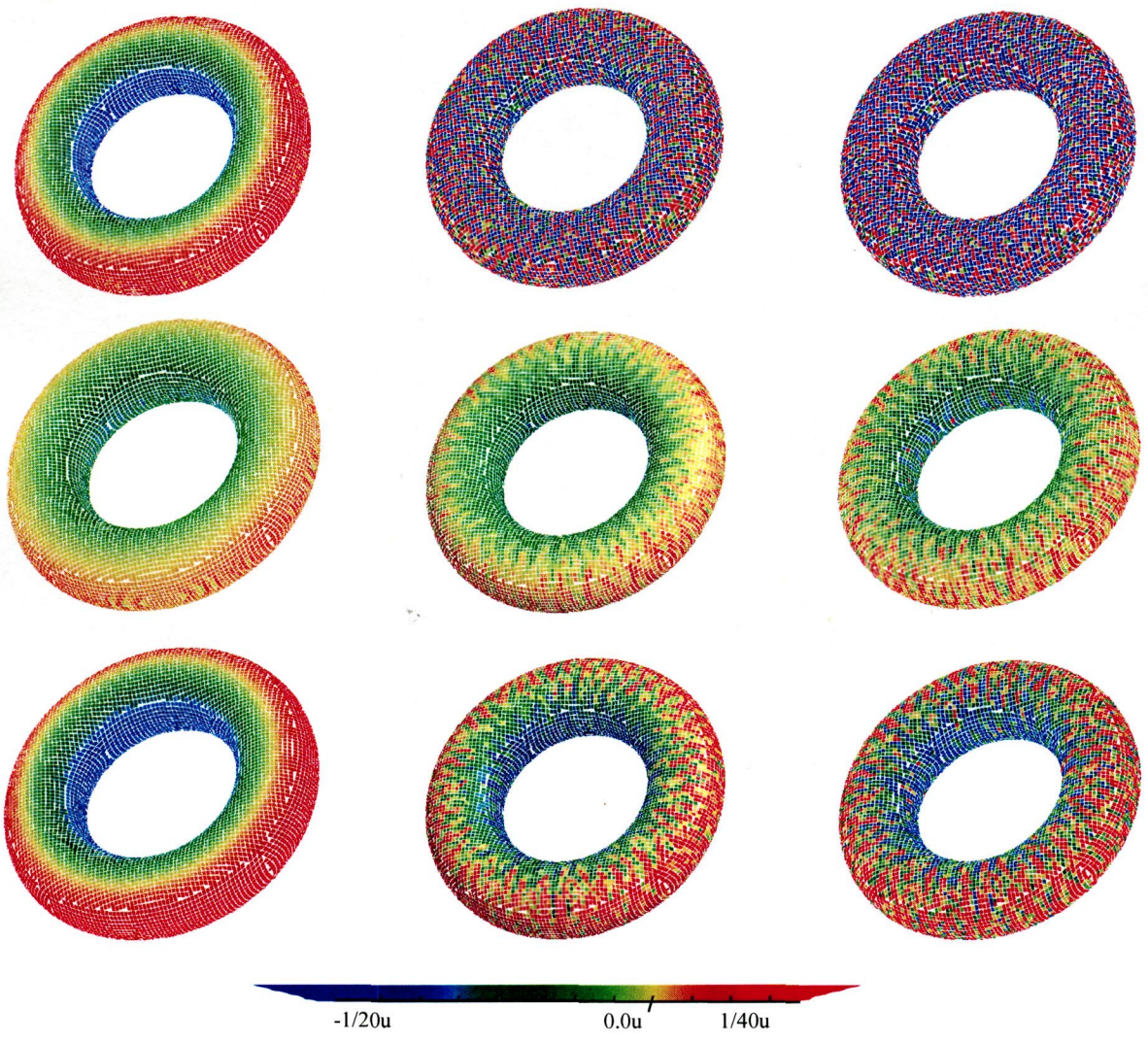


Figura 9.7: Curvatura mínima estimada sobre un toroide de radios 30 y 10, al que se ha añadido ruido de magnitud (de izquierda a derecha) 0, 5 y 10. De arriba a abajo: método cuadrático previo aplicado a la imagen original sin suavizar, método tradicional aplicado a la imagen suavizada, y método suavizado propuesto aplicado a la imagen suavizada

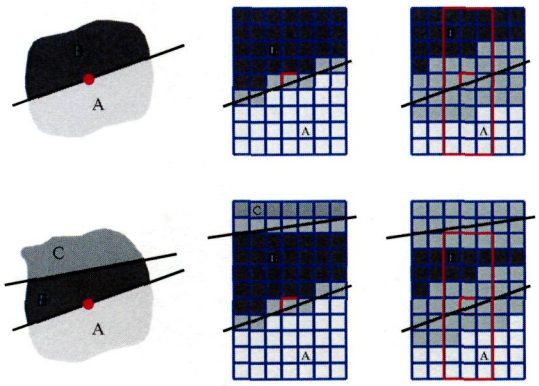


Figura 9.8: Fila superior, de izquierda a derecha: un contorno aislado, su imagen digitalizada y la imagen suavizada donde trabaja el detector. Fila inferior: dos contornos cercanos.

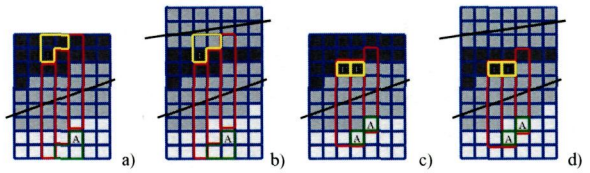


Figura 9.9: El detector propuesto estima con precisón un contorno aislado (a) pero comete error cuando hay dos contornos cercanos (b). Ajustando los límites de las columnas, el método funcionaría en ambos casos (c y d)

9.4.1 Esquema básico

Para solventar este problema, se propone no forzar a realizar las sumas de las columnas entre los mismos límites siempre, sino ajustarlos en función de cada caso. Es decir, la expresión de las sumas quedaría como sigue

$$\begin{array}{lll} S_M = S_{0,m_1,m_2,0}; & S_L = S_{-1,l_1,l_2,0}; & S_R = S_{1,r_1,r_2,0} \\ S_B = S_{0,b_1,b_2,-1}; & S_F = S_{0,f_1,f_2,1}; & S_D = S_{-1,d_1,d_2,\gamma} + S_{1,d_3,d_4,-\gamma} \end{array} \tag{9.9}$$

donde los límites de las sumatorias son tales que

$$m_1, l_1, r_1, b_1, f_1, d_1, d_3 < 0 < m_2, l_2, r_2, b_2, f_2, d_2, d_4$$

y se calculan para cada caso. ¿Cómo se calculan? Haciendo el mismo razonamiento que en la sección 6.1.1, debemos hacer llegar la sumatoria en cada columna hasta un voxel donde estimemos que hemos alcanzado el valor de la intensidad de la zona que separaba el contorno, es decir, hasta donde el valor de la derivada alcance un mínimo. De esta forma, tal y como se ve en la figura 9.9 en los casos c y d, la sumatoria acabará antes de coincidir con el área de influencia del segundo contorno.

Finalmente, para estimar las intensidades A y B se utilizarán los voxels extremos de las sumatorias, usando las columnas apropiadas en cada caso según la orientación del contorno. Por ejemplo, para el caso en el que los productos G_xG_y y G_yG_z sean negativos, las expresiones serían

$$
\begin{aligned}
b &= \frac{h_y}{2h_x\,(A-B)}\,(S_R - S_L + A\,(r_1 - l_1) - B\,(r_2 - l_2)) \\
c &= \frac{h_y}{2h_z\,(A-B)}\,(S_F - S_B + A\,(f_1 - b_1) - B\,(f_2 - b_2)) \\
d &= \frac{h_y}{2h_x^2\,(A-B)}\,(S_R - 2S_M + S_L + A\,(r_1 - 2m_1 + l_1) - B\,(r_2 - 2m_2 + l_2)) \\
f &= \frac{\gamma h_y}{2h_x h_z\,(A-B)}(S_R + S_L + S_B + S_F - S_D - 2S_M + \\
&\quad +A\,(r_1 + l_1 + f_1 + b_1 - d_1 - d_3 - 2m_1) - B\,(r_2 + l_2 + f_2 + b_2 - d_2 - d_4 - 2m_2)) \\
g &= \frac{h_y}{2h_z^2\,(A-B)}\,(S_F - 2S_M + S_B + A\,(f_1 - 2m_1 + b_1) - B\,(f_2 - 2m_2 + b_2)) \\
a &= \frac{h_y}{2\,(A-B)}\,(2S_M - A\,(1 - 2m_1) - B(1 + 2m_2)) - k\,(h_x^2 d + h_z^2 g)
\end{aligned}
$$

donde

$$k = \frac{1 + 24a_1 + 96a_2 + 96a_3}{12}$$

d) la posición y el vector normal se calculan a partir de la expresión del paraboloide

$$
\begin{aligned}
\text{corte con la vertical central del voxel} &: \quad (0, a, 0) \\
\text{vector normal} &: \quad N = \frac{A-B}{\sqrt{1 + b^2 + c^2}}\begin{bmatrix} b & -1 & c \end{bmatrix}
\end{aligned}
$$

e) las curvaturas también se calculan a partir de la expresión del paraboloide, aplicando el lema 8.1

De nuevo hay que recordar que el anterior lema sólo es aplicable para el caso $|G_y|\,h_y > |G_x|\,h_x, |G_z|\,h_z$. Cuando esto no ocurra, el esquema será idéntico, pero intercambiando las variables entre sí.

Ejemplos

Generemos una imagen compuesta por dos esferas sintéticas de radio 20, separadas por una distancia de 5 voxels. En la figura 9.11a podemos ver dicha imagen, así como una ampliación de la zona donde ambas esferan están muy cerca. Cuando dicha imagen es suavizada, la distancia entre las esferas es aún menor, como puede verse en la figura b.

Centrándonos en el voxel borde que más cerca está de la segunda esfera (mostrado en color verde), al aplicar el método de límites fijos se estará usando una vecindad demasiado lejana (mostrada en color amarillo), con lo cual habrá voxels cuya intensidad será debida a la segunda esfera, y producirá un error en los parámetros estimados en la superficie, como se muestra en la figura c, donde puede verse que, por ejemplo, la posición sub-voxel es totalmente errónea.

Sin embargo, vemos que al aplicar el método de límites variables, la vecindad utilizada no llega a alcanzar la segunda esfera, con lo cual los parámetros estimados se obtienen de forma correcta, como se muestra en la figura d.

En presencia de ruido también el funcionamiento es más preciso utilizando límites variables. En la parte izquierda de la figura 9.12 se muestra la imagen inicial con ruido añadido de magnitud 30, y el resultado con el método original del capítulo anterior. En la parte derecha se muestra el resultado después de aplicar nuestro método propuesto de límites variables a la imagen suavizada.

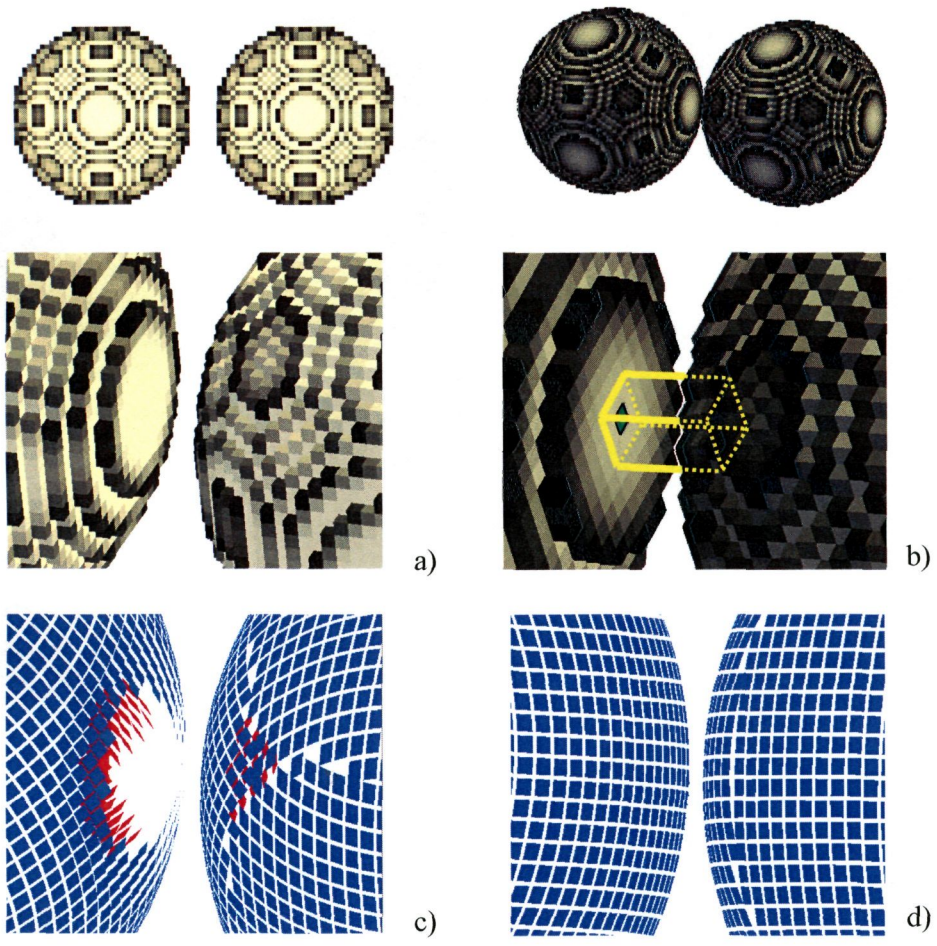


Figura 9.11: a) Imagen (y detalle) de dos esferas separadas 5 voxels; b) imagen suavizada (y ampliación detallada); c) método de límites fijos; d) método de límites variables

9.4.2 Detección de bordes muy cercanos

Si la separación entre dos contornos es demasiado cercana, la situación será como la mostrada en la figura 9.13, en donde vemos que el valor de la intensidad B se ha perdido en la imagen suavizada. Por tanto, aunque extendamos la sumatoria hasta alcanzar un mínimo en la derivada, en ese voxel el valor de la intensidad no será el correcto, y ello producirá un error.

La solución consiste, al igual que se hizo en la sección 6.1.2 para el caso bidimensional, en estimar el valor de B utilizando la imagen original en lugar de la suavizada, y generar posteriormente una subimagen sintética en donde no aparezca la influencia de ese segundo borde cercano. Recordemos en primer lugar los detalles del método para el caso 2D (figura 9.14) y luego lo extrapolaremos a tres dimensiones.

La figura 9.14a muestra muestra una "vena sintética" de tres pixels de grosor, con intensidad 200 en el interior y 100 en el exterior. La figura 9.14b muestra la imagen suavizada. Centrémonos en el pixel marcado en azul, cuyo contorno queremos estimar (sabemos que es un pixel borde porque su derivada parcial (figura

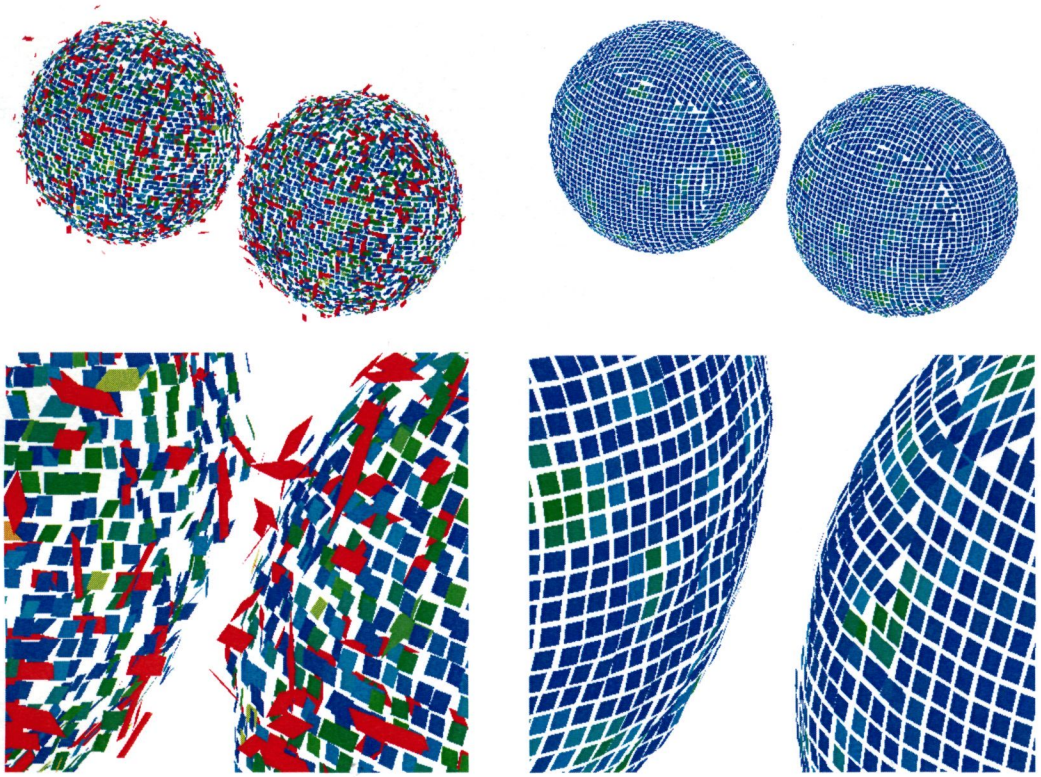


Figura 9.12: De izquierda a derecha: método de orden 2 aplicado a una imagen de dos esferas con ruido 30; método suavizado de límites variables aplicado a la imagen anterior después de suavizar

9.14c) es máxima en su columna). Con el método anterior de límites variables, utilizaríamos los límites de las sumatorias de cada columna (pixels marcados en verde en la imagen suavizada) para estimar el valor de B, lo cual produciría un error. Y sabemos que existe un segundo contorno cercano porque las derivadas parciales más allá de esos pixels tienen un valor muy alto.

La solución en este caso consiste en estimar B usando los pixels de la imagen original (pixels marcados en verde en la imagen original), y con esta estimación generar una subimagen sintética (figura 9.14d) donde todos los pixels de la parte superior son sustuidos por la estimación del valor B, eliminando así la influencia de segundo contorno cercano. Finalmente, dicha subimagen es suavizada (figura 9.14e) y sobre ella aplicamos el método anterior de límites variables.

La extrapolación del método a 3D es directa. De la misma forma, habrá que mirar más allá de los voxels donde acaban las sumatorias si las derivadas parciales alcanzan un valor muy alto. En ese caso, estimaremos B (o A si ocurre por la parte inferior) utilizando los voxels en la imagen original, y generaremos una subimagen sintética en donde desaparezca la influencia de ese segundo contorno. De hecho, podríamos encontrar una situación en otro tipo de imagen donde hubiesen contornos próximos a ambos lados simultáneamente. Por lo tanto, hay que detectar de forma separada si existe un contorno muy próximo tanto por encima como por debajo. Si no se encuentra ninguno, podemos aplicar el lema 9.3 directamente. En caso que se detecte al menos uno, habrá que generar las subimágenes F' y G', mirando qué zona es la que hay que actualizar, si la superior con el valor B, la inferior con el valor A, o ambas a la vez.

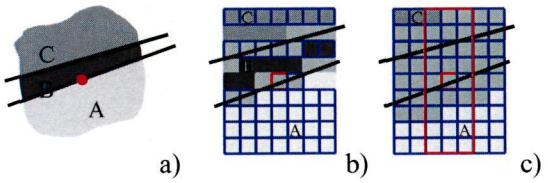


Figura 9.13: a) dos contornos muy cercanos; b) imagen digitalizada; c) imagen suavizada

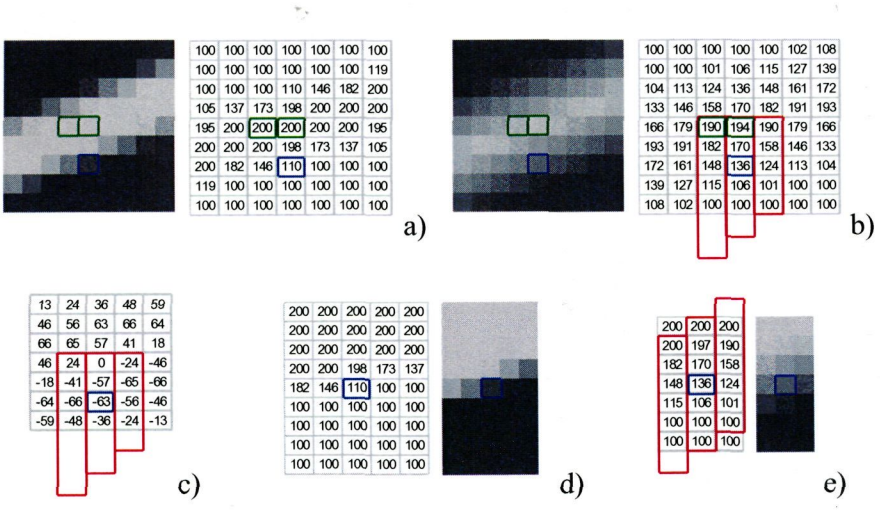


Figura 9.14:

9.4.3 Algoritmo propuesto

Todo ello queda resumido en la función que mostramos a continuación:

Funcion ContornosSuavizados3D_LimitesVariables (F, G, delta)

```
Calcular Gx, Gy, Gz
Para todos los voxels (i,j,k) de la imagen
   Si Gx(i,j,k)*Gx(i,j,k) + Gy(i,j,k)*Gy(i,j,k) + Gz(i,j,k)*Gz(i,j,k) < delta*delta Continuar
   Si (|Gx(i,j,k)|hx>|Gy(i,j,k)|hy) y (|Gx(i,j,k)|hx>|Gz(i,j,k)|hz) Entonces
      Si |Gx(i,j,k)| no es maximo en su fila Continuar
      DetectarContornos3D (i, j, k, XMAX, F, G)
   Fin Si
   Si (|Gy(i,j,k)|hy>|Gx(i,j,k)|hx) y (|Gy(i,j,k)|hy>|Gz(i,j,k)|hz) Entonces
      Si |Gy(i,j,k)| no es maximo en su columna Continuar
      DetectarContornos3D (i, j, k, YMAX, F, G)
   Fin Si
   Si (|Gz(i,j,k)|hz>|Gx(i,j,k)|hx) y (|Gz(i,j,k)|hz>|Gy(i,j,k)|hy) Entonces
      Si |Gz(i,j,k)| no es maximo en su fila Continuar
      DetectarContornos3D (i, j, k, ZMAX, F, G)
   Fin Si
Fin Para
```

Funcion DetectarContornos3D (i, j, k, orientacion, F, G)

```
Segun orientacion:
ContornoSuperior = ContornoInferior = False
Calcular voxels limites para la sumatoria
Si existe un segundo contorno muy cercano por arriba
   ContornoSuperior = True
   Estimar B a partir de la imagen F
Fin Si
Si existe un segundo contorno muy cercano por debajo
   ContornoInferior = True
   Estimar A a partir de la imagen F
Fin Si
Si ContornoSuperior o ContornoInferior
   Crear una subimagen F' centrada en (i,j,k) copiando los voxels de F
   Si ContornoSuperior actualizar B en la zona superior de F'
   Si ContornoInferior actualizar A en la zona inferior de F'
   Suavizar F' para obtener G'
   Aplicar lema 9.3 en G'(i,j,k) usando la orientacion seleccionada
Si No
   Aplicar lema 9.3 a G(i,j,k) usando la orientacion seleccionada
Fin Si
```

9.4.4 Ejemplos

Podemos rehacer el ejemplo de la figura 9.11, pero utilizando una separación menor (de un único voxel entre ambas esferas). En la figura 9.15 puede verse el resultado. Cuando la imagen sintética es ideal, la detección de

los contornos es muy precisa. Incluso añadiendo ruido, el resultado también es bastante bueno.

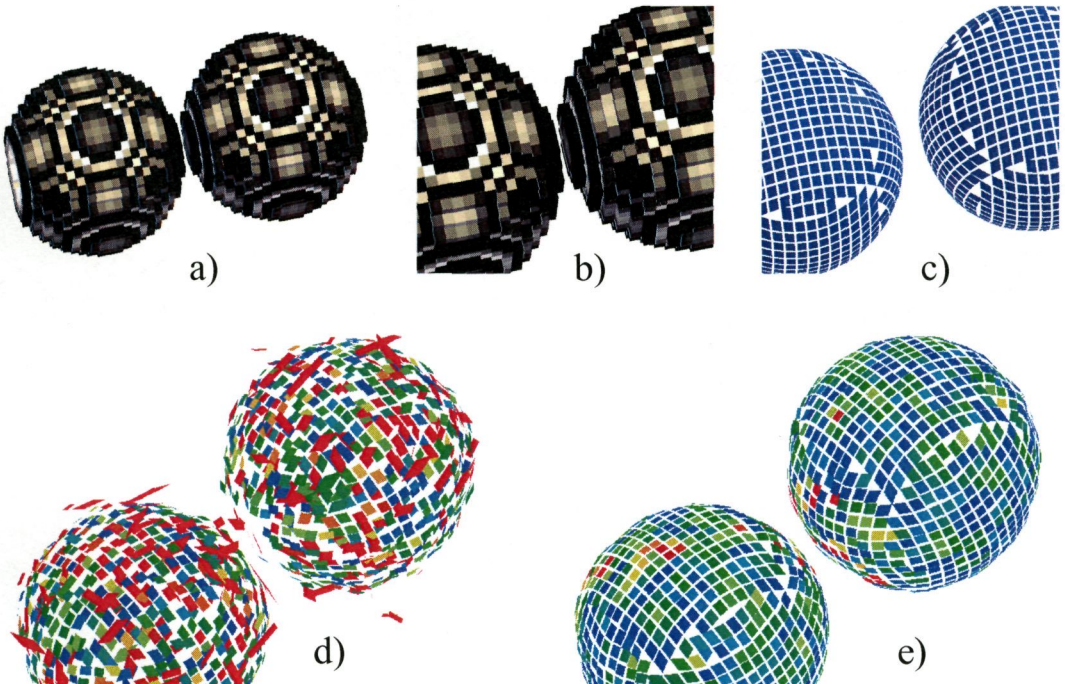


Figura 9.15: a) Imagen de dos esferas separadas 1 voxel; b) detalle de la anterior; c) contornos detectados; d) imagen con ruido añadido de magnitud 30; e) contornos detectados

9.5 Conclusiones

Se ha desarrollado un esquema capaz de encontrar en imágenes ideales sintéticas, previamente convolucionadas con una máscara de suavizado, el vector normal, los valores y direcciones principales de curvatura, y la localización sub-voxel de una superficie de segundo grado de forma exacta, así como el salto de intensidad producido a ambos lados del contorno. A partir de dicho esquema hemos desarrollado un algoritmo que detecta con precisión los contornos presentes en la imagen incluso en presencia de ruido.

Sin embargo, para poder aplicar este método con éxito a imágenes reales (concretamente imágenes angiográficas), hay que resolver dos problemas. El primero es el ruido presente en las imagenes digitalizadas, que suele ser bastante alto. Para poder eliminarlo no basta con suavizar la imagen con una máscara 3x3x3 tal y como hacemos ahora, sino que debería utilizarse un método de restauración similar al que se planteó en el caso bidimensional, de tal forma que, en sucesivas iteraciones, se vaya eliminando ruido y mejorando la calidad de la imagen, para aplicar el detector de contornos finalmente en la imagen restaurada.

El segundo problema concierne a la curvatura de los objetos (de los vasos principalmente). En una imagen pueden existir vasos de grosor muy pequeño (3 ó 4 voxels de diámetro). En estos casos, cuando la curvatura es muy alta, existe un error entre los valores obtenidos por nuestro método (el cual busca paraboloides) y los

reales (que se asemejan más a trozos de toroides). Este hecho puede verse en la figura 9.16, donde se muestran cuatro toroides sintéticos (imitando vasos reales) con grosor pequeño. Puede verse cómo cuanto menor grosor tenga el vaso (más pequeño el radio de curvatura máxima), mayor error comete nuestro método detector.

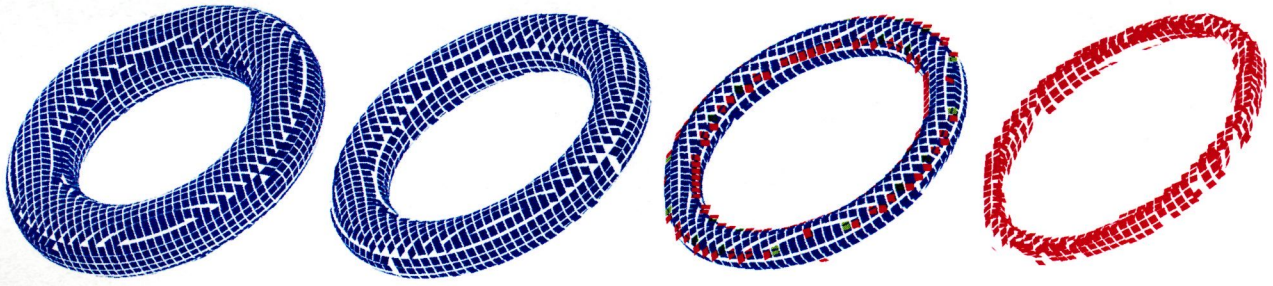


Figura 9.16: De izquierda a derecha, toroides con radio de curvatura máximo 4, 3, 2 y 1 voxels.

Este error es similar al que ocurría en el caso 2D, cuando exponíamos la diferencia existente entre una parábola y un arco de circunferencia cuando la curvatura es muy alta. Por tanto, para solventar este error, deberíamos modificar nuestro método detector para que sea capaz de estimar el trozo de toroide interior al voxel borde, en lugar de estimar un paraboloide. En el próximo capítulo propondremos un método para solventar ambos problemas.

Capítulo 10

Restauración de imágenes

- Ya sólo queda por convertir a 3D el método de restauración.
- Y hacer las pruebas.
- ¿Tendremos máquina suficiente?.....

FEBRERO 2004

Ya hemos visto que la adquisición, digitalización y manipulación de imágenes introduce un cierto número de perturbaciones en la imagen original que dificultan en muchas ocasiones su correcta interpretación. Es lo que se conoce como ruido en la imagen. Cuando la magnitud de la perturbación no es excesiva, el método desarrollado en el capítulo previo es una buena herramienta para solventar el problema. Pero cuando el ruido es grande, hay que buscar otros métodos.

En estos casos, al igual que sucedía en las imágenes bidimensionales, lo usual es definir un proceso iterativo que vaya modificando gradualmente la imagen, tratando de eliminar ruido en cada iteración, de forma que al final del proceso obtengamos una imagen más limpia que la original. Es lo que se conoce como realzado o **restauración de imágenes**. Finalmente, los parámetros del contorno se evalúan sobre la imagen restaurada.

La mayoría de los métodos propuestos son extrapolaciones de los desarrollados para imágenes 2D, con el consiguiente coste computacional implícito al pasar a una dimensión mayor. En el presente capítulo comenzaremos viendo algunas de estas técnicas tradicionales para restaurar la imagen, y a continuación expondremos dos métodos nuevos (uno básico, de primer y segundo orden, y otro optimizado) a partir del esquema desarrollado en el capítulo anterior, y luego someteremos ambos esquemas a pruebas con imágenes sintéticas y reales.

10.1 Técnicas tradicionales de restauración de imágenes

En la sección 5.1 ya se comentaron las ideas principales de los métodos de difusión, cuyo objetivo consiste en minimizar el funcional de energía siguiente:

$$E_t(f) = \int_\Omega (f - f_0)^2 + t \int_\Omega \|\nabla f\|^2$$

donde f_0 es la imagen original, y $t \geq 0$ es un parámetro que determina el balance entre los dos términos. Dicho mínimo, una vez encontrado, representará la imagen restaurada. También se demostró que encontrar dicha imagen es equivalente a resolver la siguiente ecuación en derivadas parciales:

$$(f_t - f_0) - tdiv\,(\nabla f_t) = 0 \tag{10.1}$$

donde f_t es la imagen restaurada (con un cierto factor de escala t) que estamos buscando. Esta ecuación coincide con la ecuación clásica de difusión del calor, la cual suele representarse como

$$\frac{\partial f}{\partial t} = div\,(\nabla f) = \Delta f \tag{10.2}$$

La mayor desventaja de esta ecuación es que la difusión es isotrópica, es decir, se produce en todos los puntos por igual, con lo cual los contornos de la imagen resultan difuminados, perdiendo información. De hecho, puede demostrarse que resolver la ecuación anterior es equivalente a convolucionar la imagen con un núcleo gaussiano de desviación estándar $\sigma = \sqrt{2t}$, lo cual explica el efecto de suavizado en toda la imagen.

10.1.1 Difusión anisotrópica

La solución consiste en realizar una difusión anisotrópica, inicialmente propuesta por Perona y Malik en [PER90], y extendida a tres dimensiones por Gerig y otros en [GER92]. Este proceso consigue difuminar más en las zonas homogéneas, y menos o nada en las zonas pertenecientes a contornos. Ello se consigue introduciendo una función de difusión en la ecuación 10.2 que depende de la norma del gradiente de la imagen, $H(\|\nabla f\|)$, quedando la ecuación entonces como sigue:

$$\frac{\partial f}{\partial t} = div\,(H\nabla f) \tag{10.3}$$

Esta función de difusión ha de garantizar que la difusión sea muy pequeña en las zonas donde el gradiente sea muy elevado, por ejemplo, cuando la norma del gradiente supere un cierto umbral δ, preservando así mejor el contraste de la imagen (ver figura 10.1). Sin embargo, es difícil establecer automáticamente un valor adecuado para dicho umbral. Algunos autores lo hacen basándose en el histograma acumulado del gradiente [PER90], a partir de las propiedades estadísticas de cada región [YOO93], o en función de la geometría local de la imagen [LUO94].

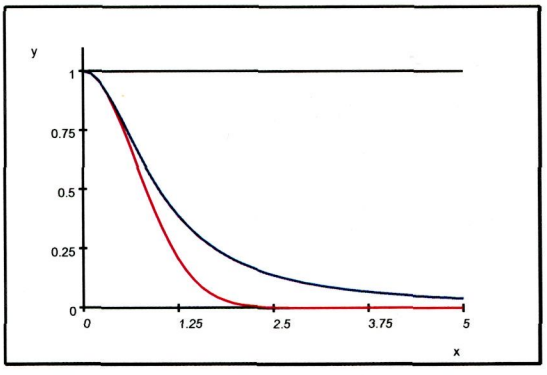


Figura 10.1: Funciones propuestas por Perona-Malik para $H(s)$, tomando $\delta = 1$: e^{-s^2/δ^2} (color rojo) y $\delta^2/\left(\delta^2 + s^2\right)$ (color azul)

Difusión matricial Otros autores, como Cottet y Germain [COT93], proponen utilizar como término de difusión una matriz D en lugar de un escalar, quedando la ecuación como sigue:

$$\frac{\partial f}{\partial t} = div\left(D\nabla f\right) \tag{10.4}$$

donde D es el operador de proyección en la dirección del gradiente:

$$D = \frac{1}{\|\nabla f\|^2}\left(\nabla f \cdot \nabla f^t\right) = \frac{1}{\|\nabla f\|^2}P$$

El efecto de esta matriz permite reorientar la dirección de la difusión en cada caso. La matriz P, también conocida como tensor de estructura, viene dada por los productos de las primeras derivadas de la imagen:

$$P = \begin{pmatrix} f_x^2 & f_x f_y \\ f_x f_y & f_y^2 \end{pmatrix}$$

y ha sido ampliamente usada para extraer la orientación local en una imagen, como por ejemplo en aplicaciones para reconocimiento de huellas dactilares [ALM00] o análisis de texturas [BIG91].

Weickert [WEI94b] propone usar una matriz que posea los mismos autovectores que el tensor de estructura anterior, previamente convolucionado con un núcleo gaussiano, lo que permite una mejor regularización. Sánchez-Ortiz [SAN99] propone otra matriz que dependa de una información conocida a priori de la imagen, y así poder incluir un conocimiento del modelo a estudiar (en su caso el estudio del movimiento del corazón).

Interpretación geométrica del proceso de difusión En el caso bidimensional ya comentamos que la difusión anisotrópica de Perona y Malik puede interpretarse como un proceso de difusión en las direcciones paralela y normal del gradiente, con distintos coeficientes de difusión, tal y como se demostró en la sección 5.1.2, a través de la ecuación

$$div\left(H\nabla f\right) = \left(H + sH'\right) f_{\eta\eta} + H f_{\xi\xi}$$

donde $f_{\eta\eta}$ y $f_{\xi\xi}$ son las segundas derivadas en la dirección paralela y normal del gradiente respectivamente. De esta forma, la función H era convenientemente elegida para difuminar más en la dirección del contorno (normal al gradiente) y menos en la dirección del gradiente.

En el caso tridimensional, la demostración de que la ecuación 10.3 es equivalente a una suma ponderada de las segundas derivadas parciales es bastante más compleja, y puede verse en [TEB98], donde se demuestra que

$$div\left(H\nabla f\right) = \left(H + sH'\right) f_{\eta\eta} + H\left(\Delta f - f_{\eta\eta}\right)$$

Esta ecuación demuestra que la ecuación de Perona y Malik es isotrópica en el plano normal al gradiente, es decir, sobre el plano tangente a la isosuperficie de la imagen.

En cuanto a la difusión matricial (ecuación 10.4), también se llega a un resultado más complejo que el anterior (ver [KRI00d]), donde la divergencia llega a expresarse en términos de segundas derivadas en las direcciones de los autovectores de la matriz de difusión, ponderadas por sus autovalores asociados, más otros términos de mayor complejidad. En el caso particular en que uno de los autovectores de la matriz de difusión tenga la orientación del vector gradiente, la difusión sería exactamente igual que la de Perona y Malik.

10.1.2 Flujo multidireccional

Un nuevo esquema de difusión propuesto por Krissian en [KRI02] permite un esquema de restauración más eficiente en imágenes tridimensionales de angiografías, ya que se centra en preservar al máximo estructuras geométricas tubulares como los vasos sanguíneos. Para ello define una base de $\mathbb{R}^3$ que depende de la estructura

local de la imagen, dada por los vectores unitarios $\{u_0, u_1, u_2\}$. El flujo de difusión para esta base puede escribirse como

$$\mathbf{F} = \sum_{i=0}^{2} \phi_i(f_{u_i}) u_i$$

y la ecuación de difusión queda de la siguiente manera

$$\frac{\partial f}{\partial t} = div\,(\mathbf{F}) + \beta(f_0 - f)$$

donde β es un coeficiente que permite al esquema converger hacia una imagen restaurada que no sea demasiado diferente de la imagen original.

La base de vectores elegida es similar a la propuesta en [KRI97], dada por $\{\eta^*, u_1^*, u_2^*\}$, donde η^* es el vector unitario en la dirección del gradiente, y u_1^* y u_2^* son las direcciones de las curvaturas máximas y mínimas respectivamente. El superíndice * representa que antes de realizar los cálculos de las direcciones, la imagen ha sido convolucionada con un núcleo gaussiano previamente. Esta base es de mucha importancia en el contexto de estructuras pequeñas y alargadas como los vasos sanguíneos, donde la mínima curvatura indica la dirección del vaso, y la máxima curvatura es relativa al tamaño de la sección del vaso, normal al gradiente (ver figura 10.2).

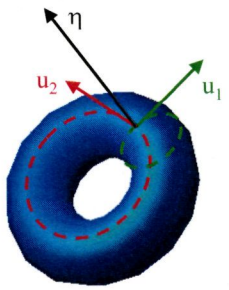


Figura 10.2: Base de vectores $\{\eta, u_1, u_2\}$, donde η es el vector normal, y u_1 y u_2 son las direcciones de las curvaturas máximas y mínimas respectivamente.

En este caso, el término de la divergencia puede descomponerse como una suma de difusiones en cada una de las direcciones de la base, de la forma siguiente:

$$div\,(\mathbf{F}) = div\left(\sum_{i=0}^{2} \phi_i(f_{u_i^*}) u_i^*\right) = \sum_{i=0}^{2} div\left(\phi_i(f_{u_i^*}) u_i^*\right)$$

Cada término de esta sumatoria puede interpretarse como una difusión anisotrópica a lo largo de la dirección de cada vector de la base, la cual a su vez puede representarse como:

$$\begin{aligned} div\,(\phi(f_u)u) &= \phi(f_u)div(u) + \phi'(f_u)\partial_u(f_u) = \\ &= \phi(f_u)K_u + \phi'(f_u)\left[f_{uu} + u^t J(u)^t \nabla f\right] \end{aligned}$$

donde K_u indica la curvatura de la isosuperficie asociada a la dirección u, y $J(u)$ es la matriz jacobiana del vector u. Esta última expresión puede interpretarse de la siguiente manera:

1. la función ϕ pesa una difusión que es proporcional a la curvatura de la isosuperficie asociada a la dirección u, la cual actúa como un promedio de la curvatura media.

2. ϕ' tiene un efecto diferente en función del signo que tenga: si $\phi' > 0$ el término ϕf_{uu} actúa como la ecuación del calor en la dirección de u generando un efecto de suavizado; si por el contrario $\phi' < 0$ entonces tiene un efecto de realce del contraste en la dirección de u.

Como la idea que se pretende es preservar estructuras tubulares como los vasos, la idea intuitiva sugiere suavizar más en la dirección del eje central del vaso que en la dirección de la sección. Llamando u_1 a la dirección de máxima curvatura, y u_2 a la dirección de mínima curvatura, las funciones ϕ_i propuestas por el autor son:

$$\begin{aligned} \phi_0(s) &= se^{-\left(\frac{s}{\delta}\right)^2} \\ \phi_1(f_{u_1^*}) &= \alpha_1 u_1^* \\ \phi_2(f_{u_2^*}) &= \alpha_2 u_2^* \end{aligned} \tag{10.5}$$

donde ϕ_0 es la misma función elegida por Perona y Malik, el cual tiene el efecto de ligero realce de los contornos, y ϕ_1 y ϕ_2 son funciones de difusión con un efecto mayor en la dirección de mínima curvatura, es decir $0 < \alpha_1 < \alpha_2 < 1$.

10.1.3 Problemas de la formulación

Los problemas asociados a este tipo de esquemas son los mismos que ya comentamos en la sección 5.1.3. Visualmente los resultados son de bastante calidad, pero no son exactos en cuanto a la precisión en la estimación de las características de los contornos (posición sub-voxel, dirección normal al contorno, y valores y direcciones de curvatura), ni siquiera en imágenes sintéticas. Esto es así debido a varios factores:

1. En primer lugar, todo el desarrollo matemático parte de que la imagen f es continua y derivable en todo su dominio, lo que contradice nuestra hipótesis de trabajo. De esta forma, aunque en el plano continuo los resultados serían exactos, no es así al trabajar con una imagen discretizada.

2. El cómputo de las primeras y segundas derivadas en las direcciones x, y y z se hace siempre a partir de la convolución con máscaras como las vistas en la sección 7.1.3, donde ya se comentó el error que producían incluso en imágenes ideales cuando las orientaciones de los contornos no coincidían con las direcciones principales.

3. Las derivadas direccionales se calculan a partir de las derivadas principales, que ya tienen una cierta medida de error. Además, sólo disponemos de valores puntuales de la imagen en el interior de una malla rectangular tridimensional orientada en las direcciones x, y y z.

4. Las direcciones en las que se realiza el proceso de difusión en cada voxel son también estimadas en base a expresiones como las vistas en la sección 8.1, todas ellas deducidas a partir de un caso continuo, pero que generan error al utilizar una imagen discretizada.

5. El esquema parte de que el valor puntual de cada voxel representa el valor de la función en un punto concreto dentro del voxel, generalmente su centro geométrico. Es decir, se asume que $F(x_i, y_j, z_k) = f(x_i, y_j, z_k)$. Sin embargo, en nuestra hipótesis de trabajo inicial, partimos de que el valor del voxel viene dado por la expresión

$$F(x_i, y_j, z_k) = \frac{1}{h_x h_y h_z} \int_{z_k - h_z/2}^{z_k + h_z/2} \int_{y_j - h_y/2}^{y_j + h_y/2} \int_{x_i - h_x/2}^{x_i + h_x/2} f(x_i, y_j, z_k) dx dy dz$$

Finalmente hay que reseñar que tanto el valor umbral de δ en la función de Perona y Malik, como los autovalores de la matriz de difusión en la ecuación 10.4 de difusión matricial, o como los coeficientes α_1 y α_2 de la ecuación 10.5 para el flujo multidireccional, son valores vitales para el resultado del proceso, con lo cual la tarea de decisión de su valor concreto no es nada fácil. Por otro lado, son coeficientes globales que afectan a toda la imagen por igual. Por tanto, puede ocurrir que para imágenes concretas no exista el valor óptimo que permita restaurar la imagen con la misma calidad en todos sus voxels interiores.

10.2 Método propuesto de restauración lineal

La idea principal subyacente en los métodos anteriores es siempre la misma: suavizar en aquellos voxels donde no existe contorno, y tratar de respetar los voxels en donde sí existe, bien realzando en la dirección normal al contorno, o bien suavizando a lo largo de su plano tangente. Nosotros proponemos un esquema que comparta esta idea, aprovechando los métodos propuestos en el capítulo anterior, de la siguiente forma:

10.2.1 Algoritmo básico

Sea F_0 la imagen original (figura 10.3a), a la cual le aplicamos una máscara de suavizado H_G para obtener la imagen suavizada G_0 (figura 10.3b)[1]. A esta nueva imagen le podemos aplicar nuestro algoritmo lineal para imágenes suavizadas. Dicho algoritmo nos devolverá información sobre los voxels en que existe un contorno (figura 10.3c), así como sus parámetros intravoxel (posición, vector normal, y salto de intensidad). Con esta información, podría generarse una nueva imagen 3D, F_1, tal que en las zonas alejadas de los voxels pertenecientes a un borde, mantengan el valor de la imagen suavizada, G_0, y en las zonas cercanas a los voxels borde, asignarles un valor calculado a partir de los parámetros obtenidos por el método para dicho voxel.

La forma para calcular estos valores se basa en la idea siguiente: sea el voxel (i, j, k) un voxel etiquetado por nuestro método como voxel borde. Esto significa que, a partir de los valores de una cierta ventana tridimensional centrada en dicho voxel[2], nuestro método ha deducido que por él atraviesa un plano $y = a + bx + cz$, dejando las intensidades A y B a cada lado del plano. Por lo tanto, podría generarse una subimagen sintética con las mismas dimensiones a partir de los parámetros estimados por nuestro método. Esta idea puede verse en la figura 10.3d.

Para calcular la intensidad de cada uno de los voxels de esa ventana, simplemente aplicamos nuestra hipótesis de partida pero a la inversa. Es decir, dado un voxel centrado en el punto (x_k, y_k, z_k), dado el plano $y = a + bx + cz$, y dados dos valores de intensidad a cada lado del borde, A por debajo y B por encima, se calcula el volumen relativo interior al voxel bajo el plano, V, tal que $0 \leq V \leq 1$, y se deduce la intensidad que tendrá, $I = VA + (1 - V)B$. En el anexo C.1 se deducen las expresiones para el cálculo del volumen V.

Aplicando este esquema, en el caso en que la imagen original fuera ideal, obtendríamos una pequeña imagen sintética que coincidiría exactamente con la original. Pero si la imagen original tuviese algo de ruido, es entonces cuando mayor interés habría, ya que la pequeña imagen sintética generada seguiría dando como resultado los mismos parámetros para el contorno que la original, pero tendría una **calidad visual superior.**

Repitiendo esto para cada uno de los voxels borde, ocurrirá que a lo largo de la superficie de un contorno, se irán generando diferentes subimágenes con solapes entre ellas. Por ejemplo, en la figura 10.3e, el voxel borde que aparece en el centro de la figura (remarcado en color verde) producirá una subimagen cuya columna central solapará con la columna derecha de la ventana generada para el voxel borde remarcado en rojo, y a su vez coincidirá con la columna izquierda de la ventana generada para el voxel borde remarcado en naranja. Por

[1] Aunque la figura muestre realmente un caso 2D, la explicación es idéntica para imágenes 3D.

[2] Como estamos usando el método de límites variables (sección 9.4), las dimensiones de la ventana son diferentes en cada caso (como máximo, la resolución será $5x11x5$). La orientación de la ventana también dependerá en cada caso de la derivada parcial con mayor valor absoluto en dicho voxel.

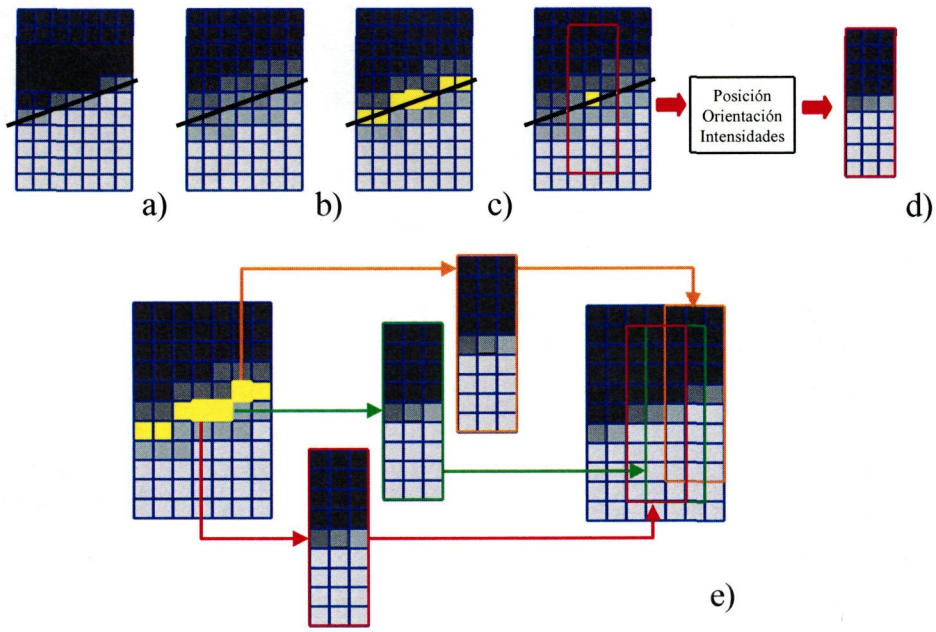


Figura 10.3: Método iterativo propuesto (caso 2D): a) Imagen original. b) Imagen suavizada. c) Pixels borde detectados por nuestro método. d) A partir de los parámetros detectados en cada pixel se genera una imagen sintética. e) Las ventanas sintéticas generadas en cada voxel borde se combinan para formar la imagen restaurada

otro lado, si suponemos que la figura muestra una sección de la imagen para un valor de x constante, también habrían solapes con columnas de las ventanas centradas en los voxels borde de las secciones contiguas en x. Para combinar toda la información y producir una única imagen F_1 proponemos el siguiente esquema:

Se crean dos imágenes de la misma resolución que la imagen original. Una de ellas es una imagen de contadores, C, donde el valor de cada voxel indica el número de ventanas que han sido generadas y en las que dicho voxel estaba incluido. La segunda imagen es una imagen de intensidades, I, donde el valor de cada voxel indica la suma acumulada de las intensidades que se han ido calculando para cada voxel. Así, cuando hayamos procesado todos los voxels borde y generado todas las subimágenes necesarias, los voxels donde $C = 0$ indicarán que están lejos de cualquier borde, y por lo tanto tendrán el mismo valor que en la imagen suavizada G_0. Por el contrario, los voxels donde $C > 0$ indicarán que están incluidos en una o más ventanas sintéticas, y por tanto su intensidad será el cociente I/C. El pseudo-código para este esquema es el siguiente, donde se hace una llamada a la función *DetectarContornos3D()* definida en la página 301.

Funcion RestaurarImagen3D (F0, F1, delta)

```
Crear una imagen de contadores C e inicializarla a 0
Crear una imagen de intensidades I e inicializarla a 0
Suavizar F con una mascara H para obtener la imagen G
Calcular las parciales Gx, Gy, Gz

Para todos los voxels (i,j,k) de la imagen G
   Si Gx(i,j,k)*Gx(i,j,k)+Gy(i,j,k)*Gy(i,j,k)+Gz(i,j,k)*Gz(i,j,k)<delta*delta Continuar
   Si (|Gx(i,j,k)|hx>|Gy(i,j,k)|hy) y (|Gx(i,j,k)|hx>|Gz(i,j,k)|hz) Entonces
```

```
      Si |Gx(i,j,k)| no es maximo en su fila Continuar
      DetectarContornos3D (i, j, k, XMAX, F, G)
      ActualizarImagenes (C, I, i, j, k, XMAX)
   Fin Si
   Si (|Gy(i,j,k)|hy>|Gx(i,j,k)|hx) y (|Gy(i,j,k)|hy>|Gz(i,j,k)|hz) Entonces
      Si |Gy(i,j,k)| no es maximo en su columna Continuar
      DetectarContornos3D (i, j, k, YMAX, F, G)
      ActualizarImagenes (C, I, i, j, k, YMAX)
   Fin Si
   Si (|Gz(i,j,k)|hz>|Gx(i,j,k)|hx) y (|Gz(i,j,k)|hz>|Gy(i,j,k)|hy) Entonces
      Si |Gz(i,j,k)| no es maximo en su fila Continuar
      DetectarContornos3D (i, j, k, ZMAX, F, G)
      ActualizarImagenes (C, I, i, j, k, ZMAX)
   Fin Si
Fin Para

Crear una imagen F1 = G
Para todos los voxels (i,j,k) de la imagen G
   Si C(i,j,k) > 0 Entonces F1(i,j,k) = I(i,j,k) / C(i,j,k)
Fin Para
```

Funcion ActualizarImagenes3D (C, I, i, j, k, orientacion)

```
Segun orientacion:
   Para todos los voxels (x,y,z) pertenecientes a una ventana centrada en (i,j,k)
      Intensidad = Calcular intensidad del voxel (x,y,z) a partir
                                 de los parametros calculados para (i,j,k)
      I(x,y,z) += Intensidad
      C(x,y,z) ++
   Fin Para
Fin Segun
```

El algoritmo anterior parte de una imagen original F_0 y produce una nueva imagen F_1 donde las zonas en las que no hay borde están suavizadas, y en las zonas cercanas a bordes, éstos aparecen más nítidos. La imagen F_1 tiene por tanto una calidad mayor, por lo cual le podríamos aplicar nuestro método para el cálculo de los parámetros y obtendríamos resultados más regulares. Esto puede verse en la figura 10.4.

Un resultado muy importante es que cuando la imagen original es un plano ideal, el método de restauración propuesto no altera la imagen, produciendo una imagen **idéntica a la original**. Esto no ocurre en ninguno de los métodos comentados en la sección anterior, ya que siempre se produce un pequeño difuminado al trabajar con los valores discretos de la imagen, que varía los valores originales de los voxels.

En el caso en que el ruido presente en F_0 fuera muy grande, entonces F_1, aún siendo mejor que F_0, podría seguir teniendo bastante distorsión. En ese caso, podemos volver a aplicar el algoritmo a F_1 para obtener una nueva imagen F_2, y así sucesivamente para obtener F_n, imagen en la cual el cálculo de los contornos debería producir resultados mucho mejores que en la original. En la figura 10.5 se muestran los resultados de varias iteraciones con la imagen ruidosa de la figura 10.4. El color de los planos obtenidos indica el error cometido en cada voxel, atendiendo a la dirección del vector normal, salto de intensidad y posición sub-voxel. Puede apreciarse como el error va disminuyendo en todas las magnitudes conforme avanzan las iteraciones.

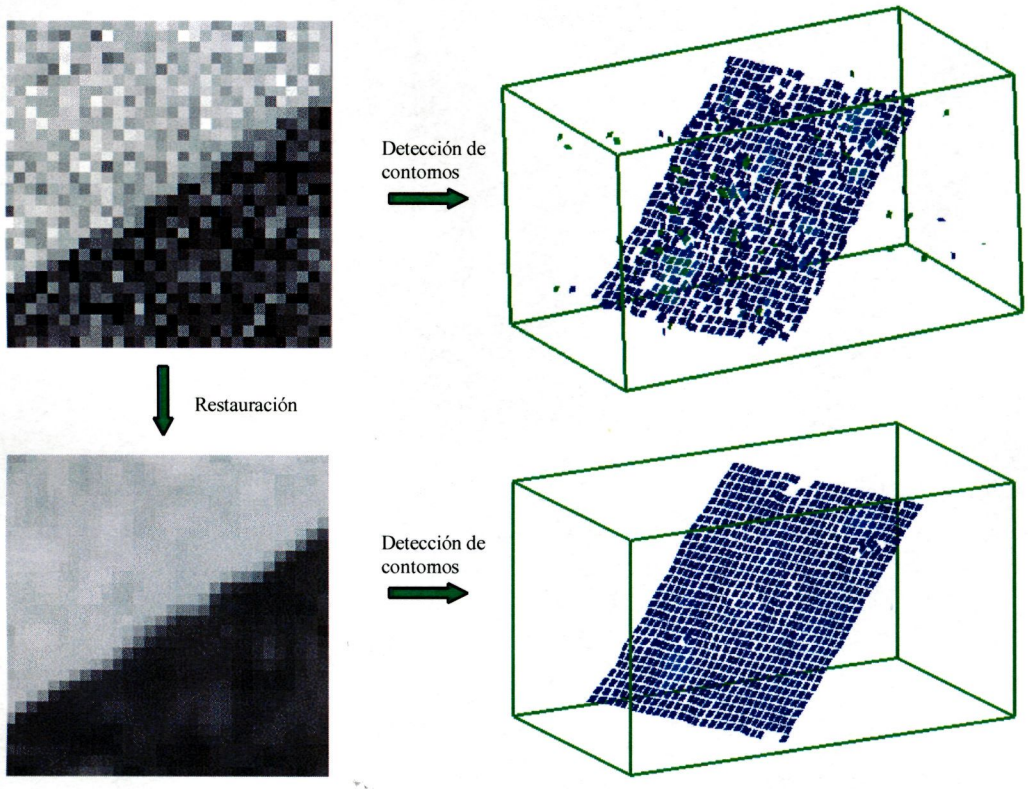


Figura 10.4: Izquierda: imagen de un contorno plano ideal con ruido añadido, y resultado de la restauración. Derecha: contornos detectados en cada imagen.

10.2.2 Aceleración de las iteraciones

Para lograr una disminución mayor del error en cada iteración hay que tener en cuenta el siguiente hecho: los parámetros obtenidos por nuestro método detector en cada voxel son para estimar las características del contorno en dicho voxel. Por tanto, aunque en la restauración hagamos que por cada voxel etiquetado como borde se genere una ventana sintética centrada en él, es la columna central de esa ventana la que más relevancia tiene, y también la que menos error comete, puesto que cualquier pequeña variación en alguno de los parámetros del contorno haría que la intensidad calculada para las columnas laterales de dicha ventana variase bastante.

Por ello, una mejora lógica sería dar más peso a la columna central que al resto de columnas en cada una de las subimágenes sintéticas que se generan. Por ejemplo, tomemos un caso simple de un contorno plano que cruza la imagen, cuyo vector normal tiene su componente y con mayor valor absoluto, como el de la figura 10.6. Después de una iteración del método de restauración, cada voxel borde pertenecerá a 9 subimágenes sintéticas: una centrada en él, con los parámetros estimados por el detector de contornos para dicho voxel, y otras 8 centradas en sus 8 vecinos.

Tal y como hemos definido el algoritmo de restauración (función *ActualizarImagenes3D*), el valor final para ese voxel será un promedio de los 9 valores estimados para dicho voxel. Sin embargo, el valor estimado por la ventana centrada en él es mucho más preciso, y debería tener un peso mayor en el promedio. De la misma manera, los 4 vecinos en las direcciones x y z (pintados en color azul en la figura 10.6) deberían tener mayor peso que los 4 vecinos de las diagonales (pintados en color rojo). El nuevo código de la función quedaría como

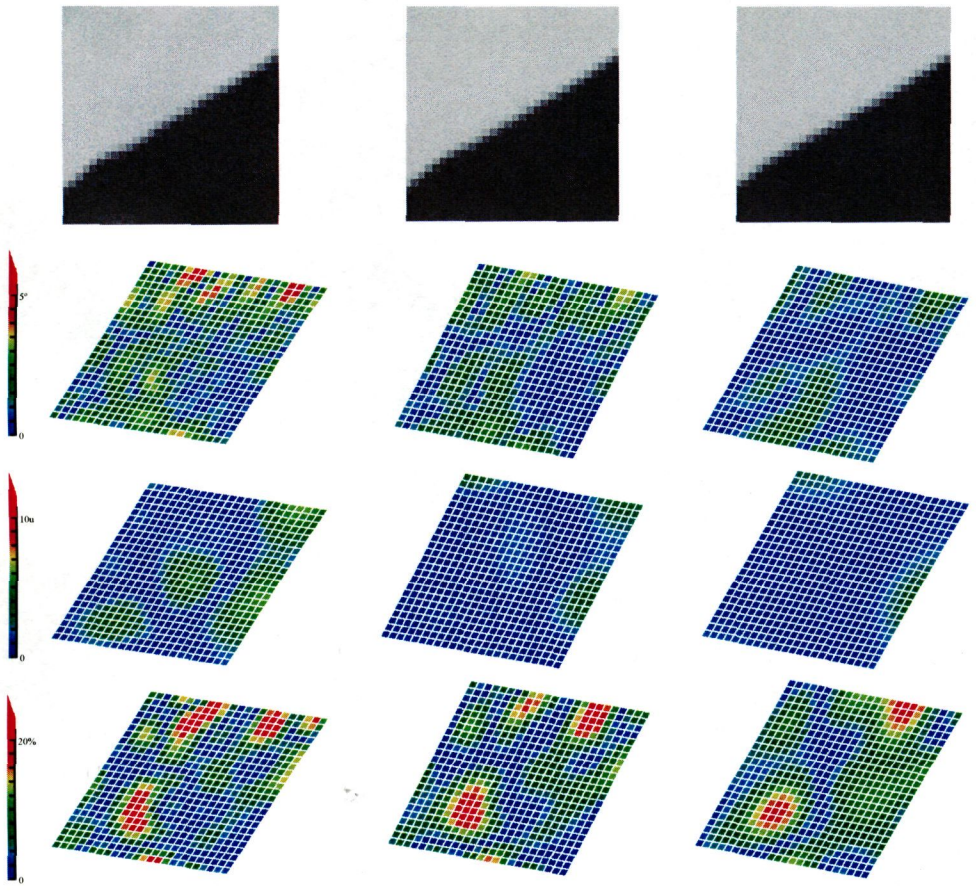


Figura 10.5: Error cometido al detectar el contorno de un plano de normal (3,5,7), con salto de intensidad de 100 unidades, y con un ruido añadido de magnitud 30, tras aplicar varias iteraciones del método de restauración (de izquierda a derecha, 10, 20 y 50 iteraciones). De arriba a a abajo: imágenes, error en la dirección normal, error en el salto de intensidad y error en la posición sub-voxel.

sigue:

Funcion ActualizarImagenes3D_ponderado (C, I, i, j, k, orientacion)

```
Segun orientacion:
   Para todos los voxels (x,y,z) pertenecientes a una ventanita centrada en (i,j,k)
      Intensidad = Calcular intensidad del voxel (x,y,z) a partir
                                 de los parametros calculados para (i,j,k)
      Si (x==i && z==k)
         I(x,y,z) += Peso0 * Intensidad
         C(x,y,z) += Peso9
      Si (x==i || z==k)
         I(x,y,z) += Peso1 * Intensidad
         C(x,y,z) += Peso2
```

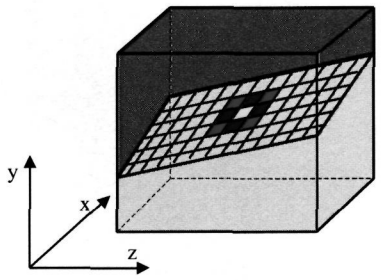


Figura 10.6:

```
        Si No
           I(x,y,z) += Intensidad
           C(x,y,z) ++
        Fin Si
     Fin Para
  Fin Segun
```

En las pruebas hemos usado los valores 100, 2 y 1 para los pesos de las columnas central, laterales y diagonales respectivamente. Podría pensarse entonces que, si el peso de la columna central es tan alto, debería entonces generarse solamente la columna central de la subimagen. Sin embargo, es interesante mantener las columnas laterales y diagonales aún con menor peso. Una de las razones es que nuestro método detector de bordes es local a cada voxel, y no garantiza que todos los voxels borde localizados a lo largo de un contorno formen una superficie continua de voxels. Dicho de otra forma, en el ejemplo de la figura 10.4, pudiera darse el caso que en alguna de las columnas, debido al elevado ruido existente, no se detectase ningún borde. Si sólo se generasen ventanas de una sola columna, esa columna donde no hubiera borde detectado se iría suavizando en cada iteración, ya que ninguna ventana caería en ella, lo que provocaría que fuese afectando al resto de columnas y finalmente toda la imagen se difuminaría.

Sin embargo, al generar ventanas de 9 columnas con un peso muy pequeño en las columnas laterales y diagonales, estamos diciendo que, si en una columna cae una ventana centrada en ella, ésta es la que decide la intensidad en dicha columna. Pero si en la columna no cae ninguna subimagen centrada, entonces se usará la intensidad estimada en otra ventana no centrada en ella que la comprenda.

Una segunda razón es que a lo largo de un contorno la orientación puede ir variando. Cada vez que la componente del vector normal de mayor valor absoluto se alterne, las ventanas pasarán de estar orientadas en una dirección principal a otra. Si sólo las generamos de grosor 1, el voxel indicado en color amarillo en la figura 10.7 no pertenecería a ninguna ventana, por lo que se iría difuminando en cada iteración. Si el contorno tocase a ese voxel, también estaríamos cometiendo un error irrecuperable en la imagen.

10.2.3 Tratamiento de los márgenes

Al igual que sucedía con las imágenes 2D (ver sección 5.2.4), es preciso establecer criterios de actuación en los voxels cercanos a los límites de la imagen (márgenes), ya que en esas zonas puede que necesitemos información de voxels vecinos que no existan. Para ello, hay que modificar el método de suavizado, y posteriormente el método de detección de contornos, en el mismo sentido en que lo hicimos para imágenes bidimensionales.

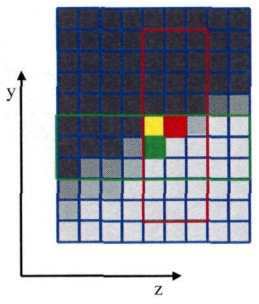


Figura 10.7: Situación de las ventanas generadas cuando un voxel (rojo) tiene un contorno con normal orientada más próxima al eje y, y su vecino (verde) con normal orientada más próxima al eje z

Suavizado

En las imágenes 3D pueden considerarse tres tipos diferentes de márgenes, tal y como se ve en la figura 10.8. Considerando los límites espaciales de la imagen como si fuese un cubo, el primer tipo (figura a) se produce cuando el voxel está pegado a una de las paredes del cubo, teniendo solamente 17 voxels vecinos en la imagen. El segundo tipo (figura b) es cuando el voxel está pegado a una de las aristas del cubo, teniendo tan solo 11 voxels vecinos. Finalmente el tercer tipo (figura c) es cuando el voxel está exactamente en la esquina del cubo, teniendo únicamente 7 vecinos.

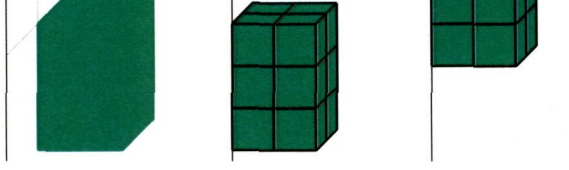


Figura 10.8: Tipos de márgenes en una imagen 3D: a) en las paredes del cubo; b) en las aristas del cubo; c) en las esquinas del cubo

Para cada tipo habrá que establecer un criterio diferente para realizar el suavizado.

Suavizado en las paredes del cubo imagen Como el suavizado se está realizando con una máscara de radio 1, aparece una incongruencia en las filas primera y última de cada dimensión, como se puede ver en la figura 10.9, donde se muestra una sección para un valor constante de x de una imagen 3D conteniendo un contorno plano que toca el margen. Una solución para estos voxels podría ser promediar solamente con sus 17 voxels vecinos (5 en la sección actual y 6 en sus secciones contiguas en x), en lugar de con los 26 vecinos como ocurre en un voxel del interior (ver figura 10.9a). Otra solución alternativa, usada en la literatura (ver [PAR97]) podría ser suponer que la imagen es contante fuera de sus márgenes, es decir, que existirían columnas virtuales por fuera de cada uno de los márgenes de la imagen, cuyo valor coincide exactamente con los voxels de cada margen (figura 10.9b).

Sin embargo, cuando se trata de un contorno que toca el margen, ninguna de estas alternativas es aconsejable,

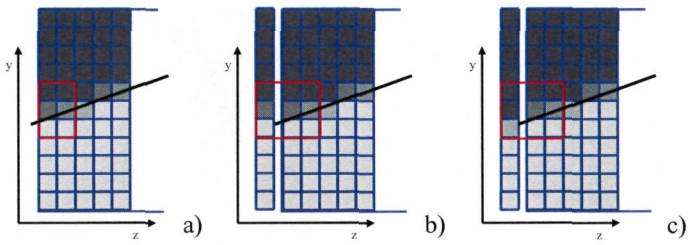


Figura 10.9: Alternativas para suavizar el margen izquierdo de una imagen: a) el voxel se promedia sólamente con sus 17 vecinos; b) se crea una columna adicional más igual a la columna del margen; c) se crea una columna adicional más con intensidades proporcionales al plano del contorno.

y es mejor elegir una tercera opción. Tomemos como ejemplo el contorno de la figura 10.9 que corta el margen izquierdo de la imagen. De no haber existido dicho margen, o de haber estado varias columnas más a la izquierda, parece lógico suponer que el plano continuaría con una orientación similar a la que tiene cuando alcanza el margen. Por lo tanto, a la izquierda del margen habrían otras tres columnas (una para cada valor de x) cuyos voxels tendrían una intensidad acorde con la posición del plano en cada uno (figura 10.9c), y que podría ser incluida dentro del proceso de suavizado.

Pero el problema es que para poder estimar entonces las intensidades de los voxels de esas nuevas columnas deberíamos conocer los parámetros del contorno en el margen, y eso aún no lo conocemos, ya que precisamente para eso es para lo que queremos suavizar la imagen. Pero lo que sí puede hacerse es aprovechar la propiedad siguiente para estimar dichas intensidades.

Sea (x_0, y_0, z_0) un voxel cualquiera del interior de la imagen, y consideremos la ventana $3x3x3$ centrada en él (ver figura 10.10). Supongamos que dicho voxel es atravesado por un plano de ecuación $y = a + bx + cz$. Llamemos $V_{i,k}$ al volumen bajo el plano interior a la columna (i,k) de dicha ventana, siendo $-1 \leq i,k \leq 1$. Suponiendo que el plano no corta el suelo ni el techo de la ventana, la expresión para el volumen es la siguiente

$$\begin{aligned} V_{i,k} &= \int_{z_0+(2k-1)h_z/2}^{z_0+(2k+1)h_z/2} \int_{x_0+(2i-1)h_x/2}^{x_0+(2i+1)h_x/2} (a+bx+cz)dxdz = \\ &= ah_xh_z + bx_0h_xh_z + cz_0h_xh_z + bih_x^2h_z + ckh_xh_z^2 \end{aligned}$$

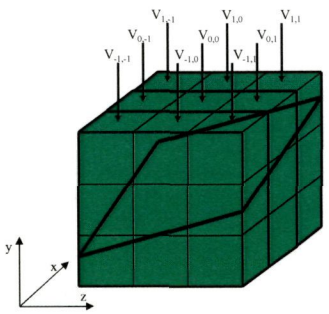


Figura 10.10: Ventana $3x3x3$ centrada en el voxel (x_0,y_0,z_0), el cual es atravesado por un plano. $V_{i,k}$ indica el volumen bajo el plano interior a cada una de las 9 columnas.

Es fácil demostrar entonces que el volumen bajo el plano interior a una de las columnas izquierda de la ventana, $V_{i,-1}$, puede expresarse en función del volumen interior a las columnas central y derecha mediante la siguiente expresión:

$$V_{i,-1} = 2V_{i,0} - V_{i,1}$$

De igual manera, llamando $S_{i,k}$ a la suma de los tres voxels de la columna $(x_0 + i, z_0 + k)$ de la imagen, se demuestra que

$$S_{i,-1} = 2S_{i,0} - S_{i,1} \tag{10.6}$$

Sea ahora H_G la máscara de suavizado $3x3x3$, cuya expresión es

$$\begin{aligned} H_G &= \left[\begin{bmatrix} a_3 & a_2 & a_3 \\ a_2 & a_1 & a_2 \\ a_3 & a_2 & a_3 \end{bmatrix}, \begin{bmatrix} a_2 & a_1 & a_2 \\ a_1 & a_0 & a_1 \\ a_2 & a_1 & a_2 \end{bmatrix}, \begin{bmatrix} a_3 & a_2 & a_3 \\ a_2 & a_1 & a_2 \\ a_3 & a_2 & a_3 \end{bmatrix}\right] = \\ &= \left[\begin{bmatrix} H_2 & H_1 & H_2 \end{bmatrix}, \begin{bmatrix} H_1 & H_0 & H_1 \end{bmatrix}, \begin{bmatrix} H_2 & H_1 & H_2 \end{bmatrix}\right] \end{aligned}$$

donde hemos llamado H_0 a la columna central de la máscara ($H_0 = [a_1, a_0, a_1]$), H_1 a las columnas laterales ($H_1 = [a_2, a_1, a_2]$), y H_2 a las columnas diagonales ($H_2 = [a_3, a_2, a_3]$). Podemos entonces afirmar que el valor del voxel (x_0, y_0, z_0) en la imagen suavizada, tras haber hecho la convolución con H_G, será el siguiente

$$G(x_0, y_0, z_0) = F(x_0, y_0, z_0) * H_G = H_0 S_{0,0} + H_1(S_{-1,0} + S_{1,0} + S_{0,-1} + S_{0,1}) + H_2(S_{-1,-1} + S_{-1,1} + S_{1,-1} + S_{1,1})$$

Utilizando la expresión 10.6 nos queda que

$$G(x_0, y_0, z_0) = S_{0,0}(H_0 + 2H_1) + (S_{-1,0} + S_{1,0})(H_1 + 2H_2)$$

Por tanto, la convolución anterior es equivalente a la siguiente convolución

$$G(x_0, y_0, z_0) = F(x_0, y_0, z_0) * \left[\begin{bmatrix} . & a_2 + 2a_3 & . \\ . & a_1 + 2a_2 & . \\ . & a_2 + 2a_3 & . \end{bmatrix}, \begin{bmatrix} . & a_1 + 2a_2 & . \\ . & a_0 + 2a_1 & . \\ . & a_1 + 2a_2 & . \end{bmatrix}, \begin{bmatrix} . & a_2 + 2a_3 & . \\ . & a_1 + 2a_2 & . \\ . & a_2 + 2a_3 & . \end{bmatrix}\right] \tag{10.7}$$

donde no se utilizan los voxels de las columnas laterales.

En realidad esta máscara de convolución es para los voxels que están tocando el plano $z = 0$. De forma simétrica se deducen las máscaras de convolución para las otras 5 paredes de la imagen.

Suavizado en las aristas del cubo imagen Cuando el voxel se encuentra en la situación de la figura 10.8b, podemos mezclar la expresión 10.7 aplicado a cada una de las dos paredes con las que limita el voxel, quedando la convolución como sigue:

$$G(x_0, y_0, z_0) = F(x_0, y_0, z_0) * k_a \left[\begin{bmatrix} \cdots \\ \cdots \\ \cdots \end{bmatrix}, \begin{bmatrix} . & 2a_1 + 4a_2 & a_2 + 2a_3 \\ . & 2a_0 + 4a_1 & a_1 + 2a_2 \\ . & 2a_1 + 4a_2 & a_2 + 2a_3 \end{bmatrix}, \begin{bmatrix} . & a_2 + 2a_3 & . \\ . & a_1 + 2a_2 & . \\ . & a_2 + 2a_3 & . \end{bmatrix}\right]$$

En este caso el factor escalar k_a que multiplica a la máscara es para que la suma de sus pesos sea uno, y su valor es

$$k_a = 1 + a_0 + 4a_1 + 4a_2$$

Éste el caso de la arista $x = z = 0$. De forma simétrica se deducen las máscaras para las otras 11 aristas del cubo imagen.

Suavizado en las esquinas del cubo imagen Finalmente, los 8 voxels de las esquinas del cubo imagen son un caso realmente particular, y tendrán muy poca relevancia en la totalidad de la imagen. En cualquier caso, aplicando de nuevo el mismo criterio para los otros tipos de margen, obtenemos la expresión

$$G(x_0,y_0,z_0) = F(x_0,y_0,z_0) * k_e \left[\begin{bmatrix} \dots \\ \dots \\ \dots \end{bmatrix}, \begin{bmatrix} . & . & . \\ . & 4a_0+8a_1 & 2a_1+4a_2 \\ . & 2a_1+4a_2 & a_2+2a_3 \end{bmatrix}, \begin{bmatrix} . & . & . \\ . & 2a_1+4a_2 & a_2+2a_3 \\ . & a_2+2a_3 & . \end{bmatrix} \right]$$

donde

$$k_e = 1 + 3a_0 + 8a_1 + 3a_2 - 2a_3$$

Esta máscara es solamente para el voxel $(0,0,y_{\max})$. Para las otras 7 esquinas se deducirían de forma simétrica.

Detección de contornos y generación de subimágenes

Una vez tenemos la imagen suavizada, aplicaremos el detector de contornos propuesto para saber en qué voxels generar las subimágenes. Tomemos como ejemplo un contorno plano cuyo vector normal tenga como componente de mayor valor absoluto la coordenada y, como el de la figura 10.11a. Ya que las subimágenes que generamos son más largas en la dirección y, no haría falta detectar nada en la columna del margen izquierdo. Simplemente haríamos la estimación para la segunda columna de la imagen, en donde dispondríamos de todos sus 26 vecinos en la imagen suavizada, y generaríamos la subimagen, la cual caería sobre la columna del margen, generando nuevos valores para la siguiente iteración. Para el margen derecho se procede de la misma manera. Esto significa que el bucle central de nuestro algoritmo sólo detectaría contornos en el interior de la imagen.

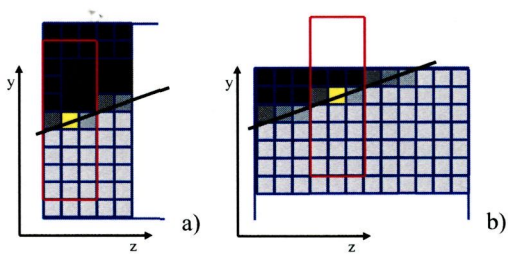


Figura 10.11: Posibles situaciones cuando un contorno plano corta un margen.

Sin embargo, cuando el contorno toca el margen de la variable y, la situación es radicalmente diferente, ya que la vecindad necesaria para realizar la estimación no puede ubicarse (figura 10.11b). En cualquier caso, alguna ventana habrá que generar en esos voxels, puesto que de lo contrario se irían difuminando en cada iteración, y traspasando este efecto al resto del contorno. La idea que proponemos es, en el caso de la figura en donde el contorno toca el margen superior, estimar solamente el valor de la intensidad bajo el contorno (eso sí puede hacerse porque la mitad inferior de la subimagen está entera) y generar una ventana con esa intensidad, dejando el resto de voxels igual que en la imagen original. Este método es exactamente el mismo que el propuesto en la sección 5.2.4 para imágenes 2D.

10.2.4 Comparación con los métodos de restauración tradicionales

Las principales diferencias entre el método de restauración propuesto y los esquemas tradicionales comentados en la sección inicial de este capítulo son prácticamente las mismas que ya se comentaron en la sección 5.2.3 para

imágenes 2D. Éstas eran:

1. **Una imagen ideal permanece inalterable.** Cuando la imagen es un plano ideal, se garantiza que $F_n = F_0$ para todas las iteraciones, sea cual sea la orientación y la posición sub-voxel del plano, y sea cual sea el salto de intensidad entre ambos lados del plano, con lo cual los parámetros que se estimen para el contorno son siempre exactos. Para los métodos de restauración tradicionales, esto no suele ser así en todos los casos: la precisión depende mucho de la orientación del plano.

2. **Disminución efectiva del ruido.** El comportamiento ante el ruido también es bastante efectivo con nuestro método, incluso para ruidos de magnitud elevada. Aunque para lograr una alta precisión a nivel numérico se necesiten muchas iteraciones, a nivel visual ya se aprecia una mejora espectacular de la imagen con un número mucho menor de iteraciones. Sin embargo, los esquemas tradicionales tratan de seguir un compromiso entre suavizar muy fuerte para eliminar el ruido pero sin difuminar demasiado el contorno, lo que los hace más lentos.

3. **Autoenfoque.** Aunque en imágenes 3D no puede hablarse de imágenes desenfocadas, es cierto que algunos contornos pueden estar ligeramente difuminados en el interior de la imagen, debido a movimientos de los objetos dentro de la imagen, o a errores en la adquisición. Si este difuminado no es excesivamente fuerte, el método propuesto es capaz de seguir encontrando los parámetros del contorno con bastante precisión. Esto significa que, como nuestro método iterativo va generando pequeñas ventanas sintéticas a partir de los parámetros obtenidos por el método detector, ya desde la primera iteración el efecto de difuminado desaparecería, apareciendo un contorno nítido tal y como presupone nuestra hipótesis inicial de trabajo.

4. **Robustez frente a niveles de ruido y de intensidad.** Para aplicar con éxito los esquemas tradicionales hay ciertos coeficientes (el umbral δ en la función de Perona y Malik, los coeficientes α_1 y α_2 de la ecuación 10.5 para el flujo multidireccional, o los autovalores de la matriz de difusión en la ecuación 10.4 de difusión matricial) que deben ajustarse a priori para cada imagen, en función del nivel de ruido presente en ella, y de la magnitud de intensidad mínima para considerar un contorno. Aún así, si en una misma imagen hubiera contornos con distintos saltos de intensidad y niveles de ruido, donde algunos estuvieran más difuminados que otros, no podría obtenerse un resultado igual de bueno para todos ellos. Por el contrario, nuestro método tiene un único parámetro que indica el umbral de intensidad. Dicho umbral no debe ser muy alto, ya que los contornos cuyo salto de intensidad sea inferior quedarán difuminados en el proceso. Sin embargo, cuando el umbral es bajo, no es tan importante el valor concreto que tenga, ya que el resultado final será bastante parecido.

10.2.5 Pruebas con imágenes sintéticas

Apliquemos el método de restauración propuesto a varias imágenes sintéticas, a las que previamente se les ha añadido ruido. En la figura 10.12a se muestra la imagen de una esfera ideal a la que se le ha añadido ruido. La figura 10.12c muestra la imagen restaurada después de 10 iteraciones. La figura 10.12d muestra a la izquierda los contornos detectados en la imagen con ruido sin restaurar, y a continuación muestra los contornos detectados sobre las imágenes resultantes de varias iteraciones intermedias. Puede observarse como visualmente la superficie del objeto se ha recuperado casi totalmente.

En la figura 10.13 hemos repetido el mismo proceso pero con un toroide, y puede observarse también como las superficies detectadas en la imagen restaurada final son bastante satisfactorias.

10.3 Método propuesto de restauración cuadrático

Las imágenes anteriores eran objetos que, aún siendo curvos, localmente a cada voxel podía considerarse que la superficie se aproximaba bastante a un plano. Sin embargo, para superficies con curvatura mayor, la aprox-

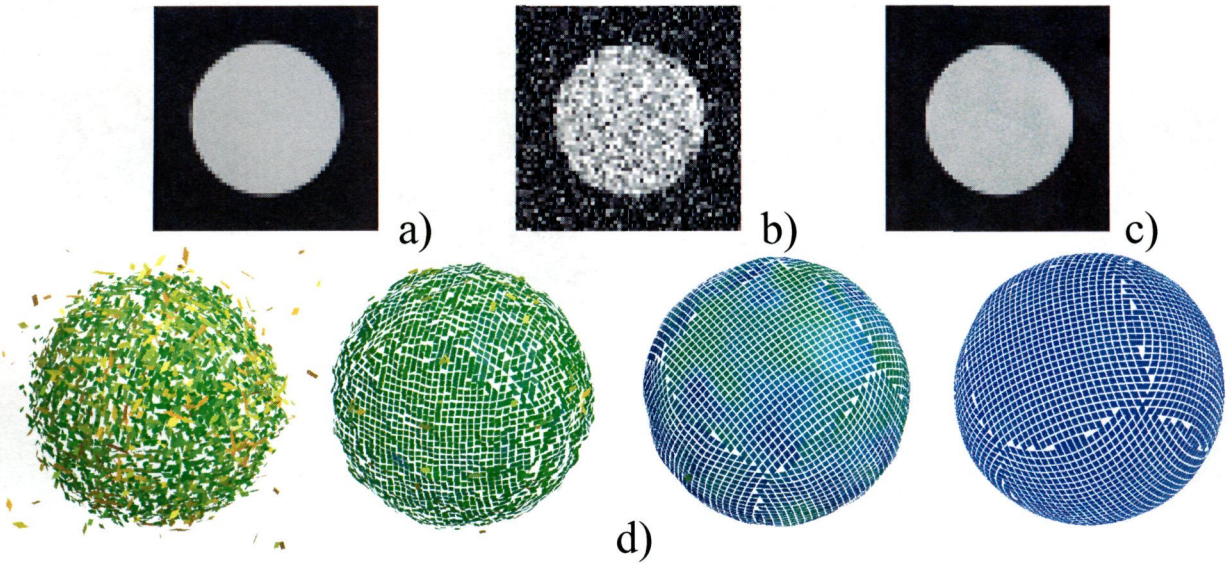


Figura 10.12: a) Imagen de una esfera ideal de radio 20, intensidad 0 en el exterior y 255 en el interior. b) Igual pero con ruido de magnitud 100 añadido. c) Imagen restaurada tras 10 iteraciones. d) De izquierda a derecha, contornos detectados en las imágenes resultantes de las iteraciones 0, 1, 5 y 10

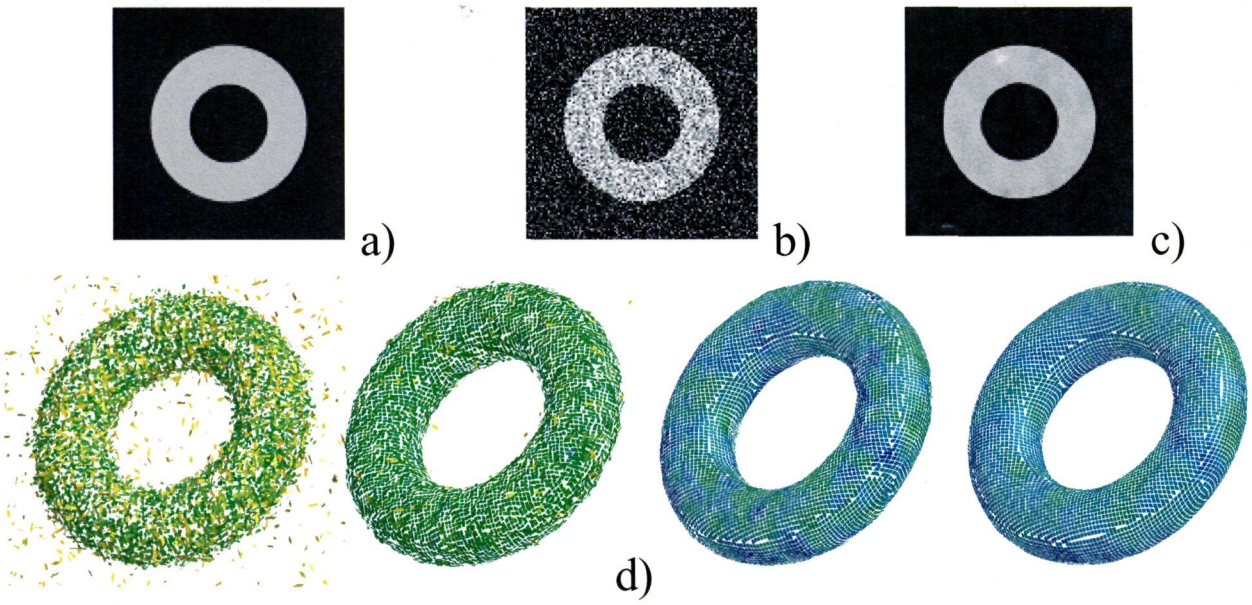


Figura 10.13: a) Imagen de un toroide ideal de radios 30 y 10, intensidad 0 en el exterior y 255 en el interior. b) Igual pero con ruido de magnitud 100 añadido. c) Imagen restaurada tras 10 iteraciones. d) De izquierda a derecha, contornos detectados en las imágenes resultantes de las iteraciones 0, 1, 5 y 10

imación lineal comienza a dar errores, como puede verse en la figura 10.14, donde la esfera va reduciendo su volumen. Esto es debido a lo siguiente:

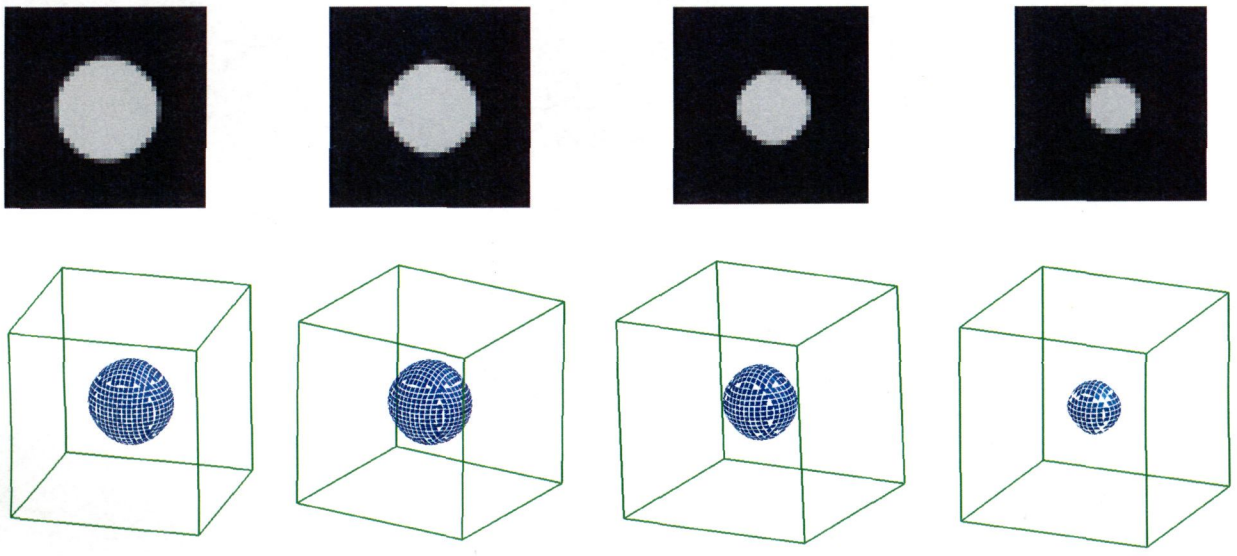


Figura 10.14: Imagen de una esfera de radio 10, y sus contornos detectados. De izquierda a derecha, resultado tras 0, 20, 40 y 60 iteraciones

Vimos en la sección anterior que, para un voxel borde con un contorno cuya normal está más próxima al eje y que a los otros ejes, el método lineal encontraba la expresión del plano $y = a' + b'x + c'z$, mientras que el método cuadrático encontraba el paraboloide $y = a + bx + cz + dx^2 + fxz + gz^2$, donde la relación entre los coeficientes de ambos métodos era

$$\begin{aligned} a &= a' - k(h_x^2 d + h_z^2 g) \\ b' &= b \\ c' &= c \end{aligned}$$

donde $k > 0$ dependía de los coeficientes de la máscara de suavizado. Es decir, si asumimos que el contorno viene dado por una expresión de segundo grado, al aproximarlo por un plano estamos cometiendo un error en la posición proporcional a los coeficientes de x^2 y z^2. Dicha posición viene medida en la vertical central del voxel ($x = z = 0$).

Por ejemplo, tomemos la imagen de la esfera de la figura 10.14, y supongamos que se ha escogido una máscara de suavizado con todos sus coeficientes iguales ($a_i = 1/27$), con lo cual $k = 3/4$ (ver 9.8). Supongamos también que cada voxel mide 1 unidad de lado ($h_x = h_y = h_z = 1$). En ese caso, puede demostrarse que en el voxel más alto en la esfera, la diferencia en vertical entre el plano y el paraboloide estimado en dicho voxel es de $k(d + g) = 0.075$ unidades, lo que supone un 7.5% del tamaño del voxel (ver figura 10.15a).

Si probásemos otro voxel donde la orientación no estuviese tan paralela al eje y (por ejemplo, la que forme un ángulo cercano a 45^o con el eje y) se vería que la diferencia en vertical entre el paraboloide y el plano sería de 0.159 unidades, es decir, prácticamente el 16% del tamaño del voxel (ver figura 10.15b). Esta diferencia con respecto al voxel anterior es debido a que, aunque la curvatura sea la misma en ambos lugares, cuanto menos vertical sea el vector normal, mayor serán los coeficientes d y g. Son estos pequeños errores los que producen

que en cada iteración, las superficies convexas se vayan aproximando hacia el interior, acabando por hacer desaparecer los objetos.

Algoritmo de segundo orden Para arreglar el problema descrito anteriormente y no perder la forma original del contorno seguiremos usando el mismo algoritmo de la sección anterior, sustituyendo el método de detección lineal por el de segundo grado. Todo lo demás (mayor peso en la columna central y tratamiento de los márgenes) sigue exactamente igual.

Simultáneamente, habrá que modificar el proceso de generación de las subimágenes para poder representar paraboloides en lugar de planos. Para ello es necesario desarrollar un método que estime la intensidad de cada voxel de esa ventana. Es decir, dado un voxel centrado en el punto (x_k, y_k, z_k), dado el paraboloide $y = a + bx + cz + dx^2 + fxz + gz^2$, y dados dos valores de intensidad a cada lado del borde, A por debajo y B por encima, se debe calcular el volumen relativo interior al voxel bajo el paraboloide, V, tal que $0 \leq V \leq 1$, y deducir la intensidad del voxel mediante la expresión $I = VA + (1 - V)B$. En el anexo C.2 se muestra un método sencillo para el cálculo del volumen V.

Aplicando este nuevo algoritmo a la imagen de la esfera de la figura 10.14 podemos observar cómo ahora la esfera no cambia de tamaño por muchas iteraciones que hagamos, y el resultado es bastante bueno incluso en presencia de ruido, como se observa en la figura 10.16. En las figuras 10.17 y 10.18 también se muestra el resultado de restaurar otros objetos sintéticos con ruido añadido, y el resultado también es bastante aceptable.

Pruebas con imagen real Probemos con una angiografía real, como la de la figura 10.19, a la cual se le han aplicado 10 iteraciones con el método cuadrático. En la fila central de dicha figura pueden verse los contornos detectados en algunas zonas de la imagen original, y en la fila inferior los contornos detectados en las mismas zonas pero sobre la imagen restaurada. Puede apreciarse como los resultados son más precisos en esta última.

Cuando se utiliza un método de restauración tradicional, la técnica más usada para visualizar el resultado suele ser aplicar un esquema de visualización de iso-superficies, del estilo del método Marching-Cubes (ver [WAT98]). Este método está basado en la idea siguiente: se supone que la imagen tridimensional es la discretización de una cierta función $f(x, y, z)$ continua y derivable. Si se fija un valor de nivel k, entonces se trata de buscar los cortes de la superficie de nivel $f(x, y, z) = k$ con las paredes de cada voxel, y generar una malla poligonal a partir de dichas intersecciones.

El problema principal consiste en la dificultad de establecer un criterio automático para la elección del valor del nivel k, ya que en función del valor obtendremos superficies diferentes. Cuando no se requiere excesiva precisión, el valor es elegido visualmente hasta encontrar una superficie aceptable. En la figura 10.20 se muestra las distintas superficies de nivel obtenidas para tres valores diferentes de k en una de las zonas que se usaron en la figura 10.19. Aparentemente el valor idóneo sería $k \approx 1000$. Sin embargo, si hubiese que dar una medida muy precisa de la localización del vaso, o una estimación de su grosor, no tendríamos una medición acertada.

Por el contrario, cuando hemos aplicado nuestro método propuesto, no sólo hemos obtenido una medida de la localización exacta de la superficie en el interior del voxel, sino que también obtenemos de forma precisa estimaciones para la dirección del vector normal, los valores y direcciones de curvatura, y también la diferencia de intensidad a ambos lados del borde.

10.4 Método propuesto de restauración para altas curvaturas

Cuando los contornos que se quieren detectar tienen curvatura elevada, como sucede por ejemplo en la superficie de los vasos, donde la curvatura máxima es igual a la inversa del radio de su sección, ocurrirán dos errores diferentes que restarán precisión a nuestro método de restauración. Analicemos primero dichos errores y finalmente veamos cómo podemos solucionarlos.

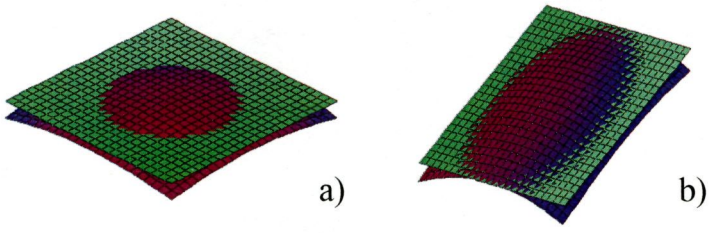


Figura 10.15: Contornos de primer y segundo orden (colores verde y violeta respectivamente) detectados sobre la imagen de una esfera centrada en $(30, 30, 30)$ de radio 10. a) en el voxel $(30, 40, 30)$; b) en el voxel $(30, 37, 37)$

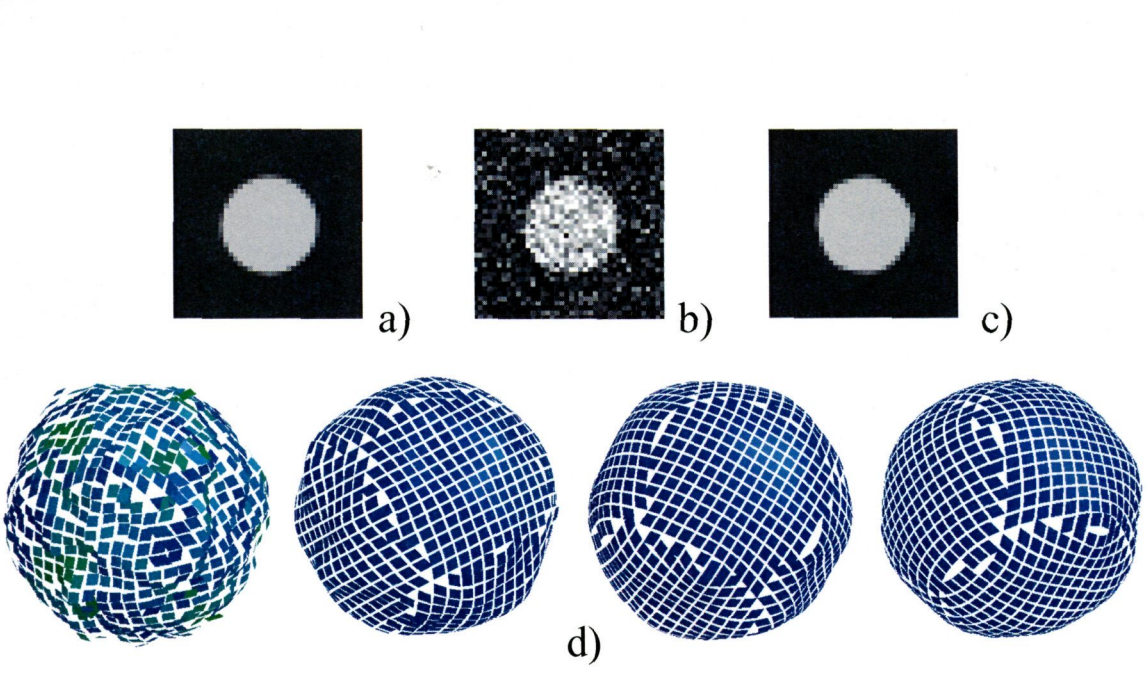


Figura 10.16: a) Imagen de una esfera ideal de radio 10, intensidad 0 en el exterior y 255 en el interior. b) Igual pero con ruido de magnitud 100 añadido. c) Imagen restaurada tras 20 iteraciones. d) De izquierda a derecha, contornos detectados en las imágenes resultantes de las iteraciones 1, 5, 10 y 20

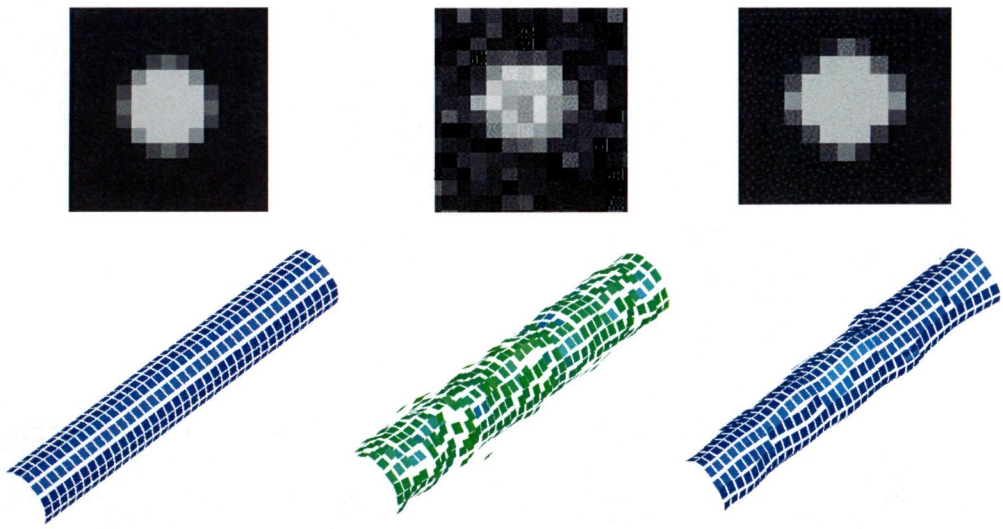


Figura 10.17: Restauración de un cilindro de radio 3 con ruido añadido de magnitud 50. De izquierda a derecha: imagen ideal, imagen con ruido, y restauración tras 10 iteraciones

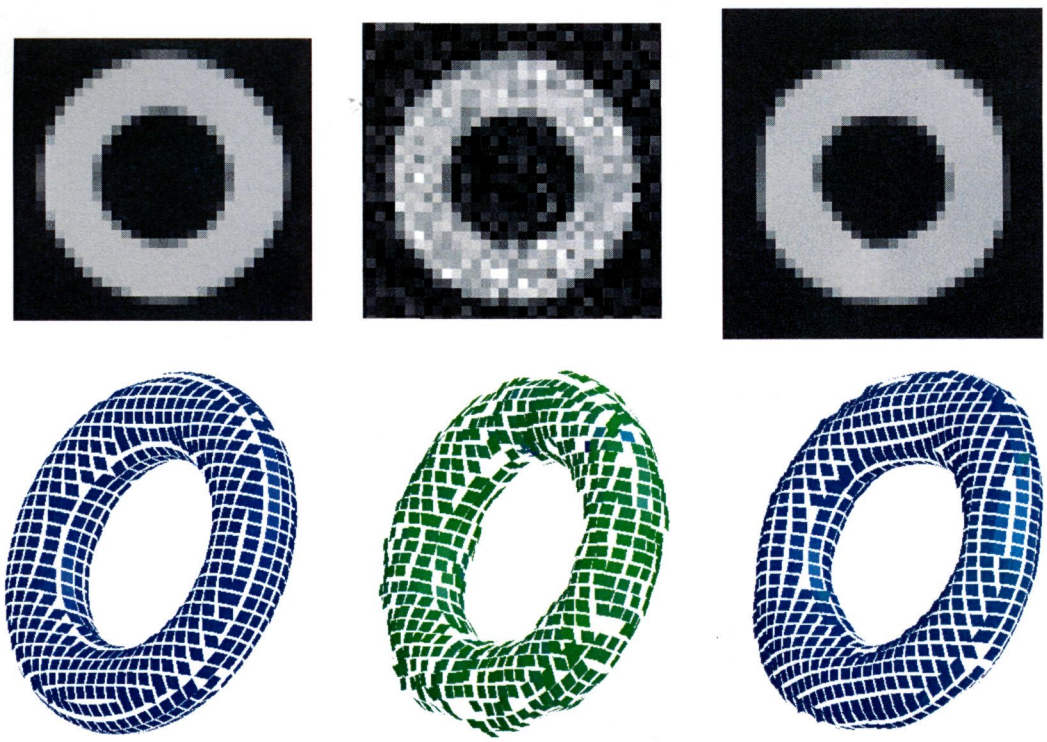


Figura 10.18: Restauración de un toroide de radios 10 y 3 con ruido añadido de magnitud 50. De izquierda a derecha: imagen ideal, imagen con ruido, y restauración tras 10 iteraciones

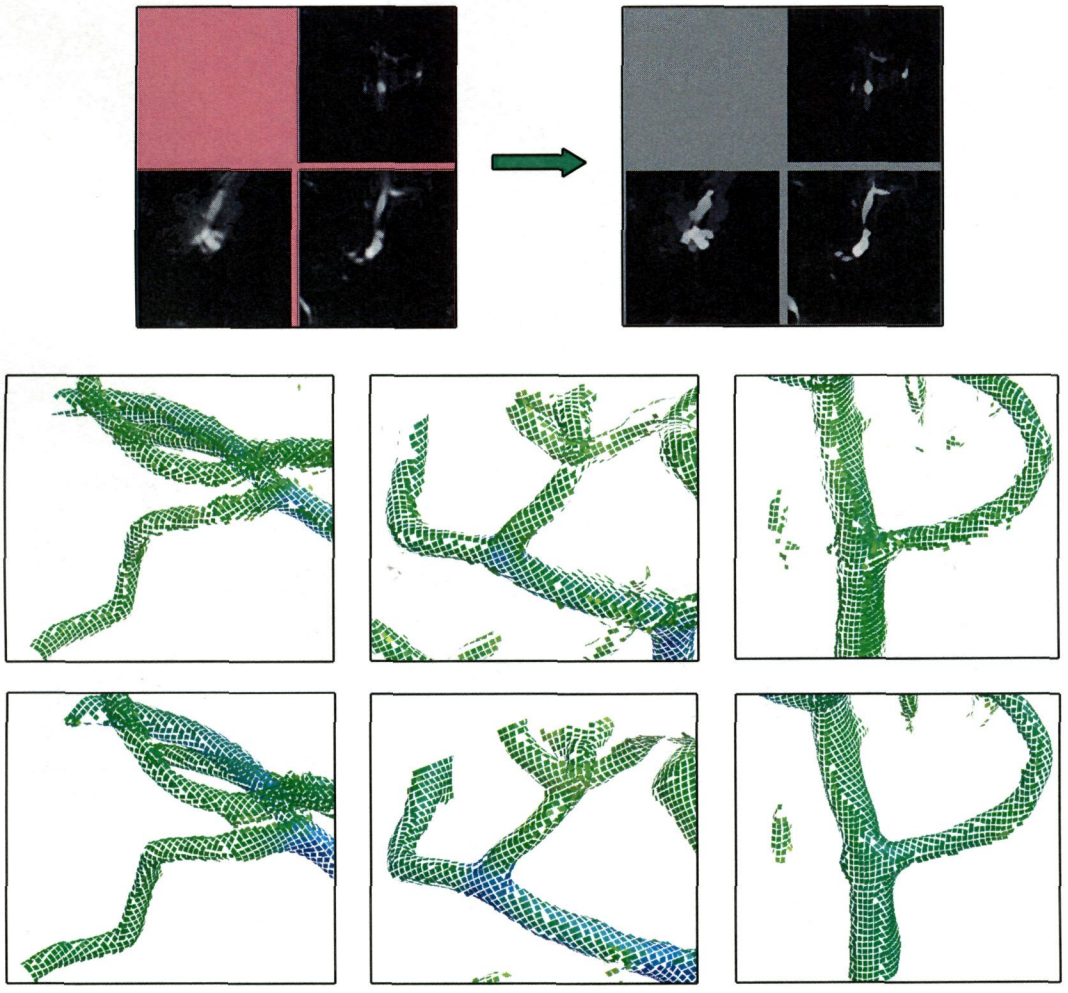


Figura 10.19: Fila superior: angiografía real y resultado de la restauración. Resto de filas: detalle de contornos detectados en la imagen original (fila central) y en la restaurada (fila inferior)

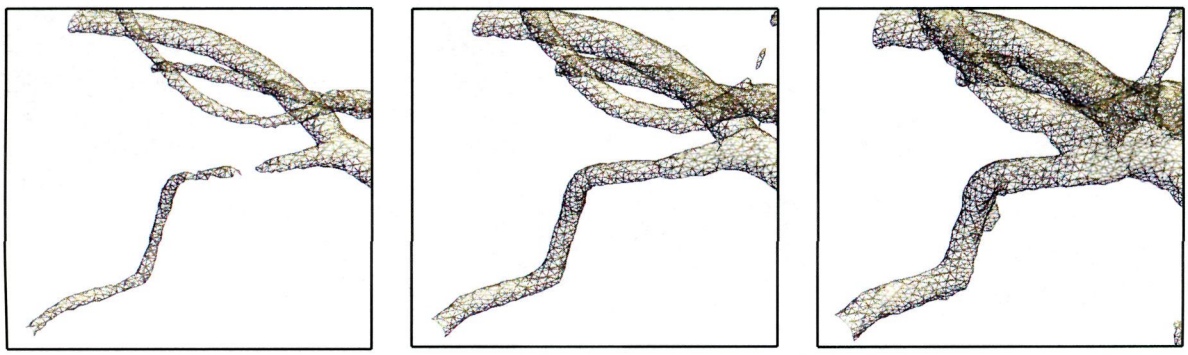


Figura 10.20: De izquierda a derecha, superficies de nivel 500, 1000 y 1500. El rango de intensidades de la imagen se encuentra entre los valores 0 y 3500

10.4.1 Errores obtenidos en superficies con alta curvatura

El primero de los errores es el cometido por nuestro método detector al aproximar la superficie del contorno por un paraboloide, es decir, por la superficie más sencilla de grado dos. Este error será más alto cuanto mayor sea la curvatura máxima de la superficie. Esto puede verse en la figura 10.21, donde se muestra sobre una esfera de radio 10 el error cometido en la estimación de la dirección normal sobre cada voxel de la superficie, primero usando nuestro método detector propuesto en el capítulo 8, y luego el detector del capítulo 9 para aplicar en imágenes previamente suavizadas.

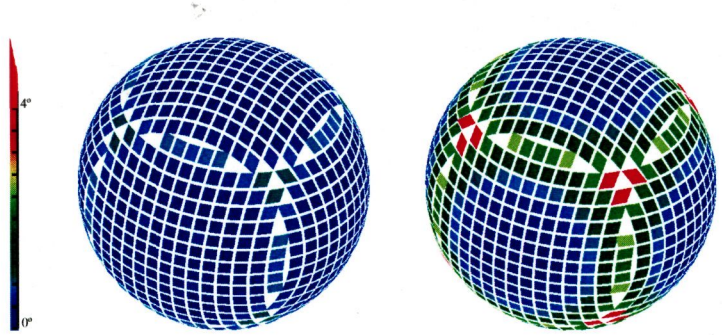


Figura 10.21: Error cometido en la estimación de la dirección normal sobre una esfera de radio 10. De izquierda a derecha: usando el método de segundo grado del capítulo 8, y usando el método de segundo grado para imágenes suavizadas

Puede apreciarse como las zonas con normal cercana a los ejes principales son aquéllas en donde se consigue mayor precisión. Sin embargo, las zonas más cercanas a las bisectrices de los planos principales son las más afectadas, puesto que en esos lugares es en donde mayor diferencia existe entre ambos objetos geométricos. Es lo mismo que ocurría en el caso 2D, cuando comentábamos la diferencia entre arcos de parábola y de circunferencia. Con el primer método el error llega a ser de 1 grado en esas zonas. Con el segundo método (para imágenes suavizadas) el error es aún mayor, llegando a ser de 4 grados en esos voxels. Esto es debido a que en el segundo método, la vecindad usada en cada voxel para la estimación de los parámetros del contorno (considerando los valores de la imagen original) es bastante mayor que con el primer método, con lo cual la diferencia entre la

superficie y un paraboloide es aún más considerable.

El segundo error, muy relacionado con el primero, consiste en la generación de subimágenes sintéticas en cada iteración de la restauración. Aunque lográsemos que el método detector obtuviese con precisión los parámetros del contorno, como nuestro método acaba por generar imágenes sintéticas de paraboloides, no será exactamente igual la imagen sintética que la original. Por ejemplo, tomemos una esfera de radio 10 como la de la figura anterior, centrada en el punto $(0,0,0)$, por lo que su expresión es

$$y = \sqrt{100 - x^2 - z^2}$$

El paraboloide vertical que comparte normal y curvaturas en el punto más alto $(0,10,0)$ tiene como expresión

$$y = 10 - \frac{1}{20}x^2 - \frac{1}{20}z^2$$

Si mostramos conjuntamente ambas superficies (figura 10.22a) vemos que existe una ligera diferencia, con lo cual al generar una subimagen sintética centrada en dicho voxel tomando dicho paraboloide, y calculando las intensidades de cada voxel en función del volumen bajo la superficie, éstas no van a coincidir con las intensidades de la imagen original. El error es aún mayor en zonas cuya normal se aleje de los ejes principales. Por ejemplo, en el punto $(0, 5\sqrt{2}, 5\sqrt{2})$, el paraboloide resultante es

$$y = \frac{1}{20}\left(100\sqrt{2} - \sqrt{2}x^2 + 20z - 2\sqrt{2}z^2\right)$$

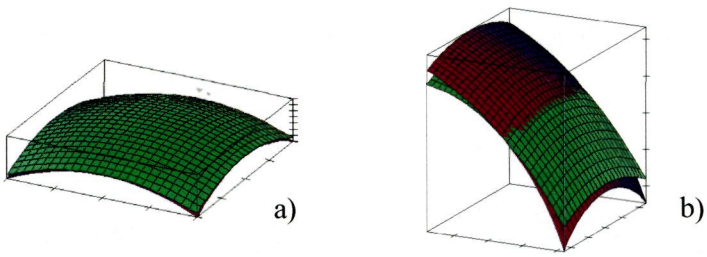


Figura 10.22: Diferencia entre una esfera de radio 10 centrada en el origen y el paraboloide que comparte normal y curvaturas en el punto: a) $(0,10,0)$; b) $(0, 5\sqrt{2}, 5\sqrt{2})$

En la figura 10.22b se aprecia la diferencia entre ambas superficies. Estas diferencias serán mayores además cuanto más altas sean las curvaturas de la superficie que se quiere detectar. Por tanto, estos errores provocan que con cada iteración del método de restauración, la superficie del objeto se vaya deformando. Esto puede verse en la figura 10.23 donde se ve el resultado de restaurar un cilindro de radio 5 y una esfera de radio 10 después de un número alto de iteraciones. Las imágenes resultantes muestran unos objetos que han sido aplanados por las zonas cuya normal se acerca a los ejes principales (ver en la figura 10.22a cómo el paraboloide se encuentra ligeramente por encima de la superficie, y por tanto puede considerarse más plana), y más afilados por las zonas cercanas a las bisectrices de los planos principales (ver en la figura 10.22b cómo el paraboloide tiende a mantener la horizontal, al contrario que la superficie que desciende más rápidamente).

10.4.2 Aproximación por toroides

¿Cuál debería ser el objeto geométrico ideal que deberíamos estimar, tanto en el método detector, como en el método de restauración en el momento de crear las subimágenes sintéticas?. En el caso bidimensional quedó

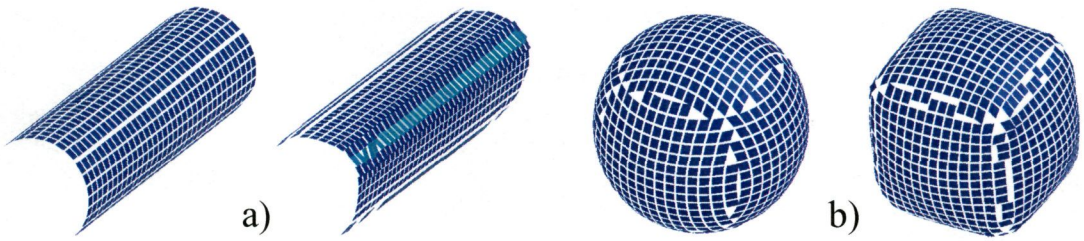


Figura 10.23: Resultado tras 50 iteraciones aplicados a la imagen de: a) un cilindro de radio 5; b) una esfera de radio 10

claro que la curva ideal es un arco de cincurferencia. Ésta es invariante frente a rotaciones, puesto que mantiene la misma curvatura en todos sus puntos independientemente de su orientación, al contrario de lo que sucede con una parábola.

En el caso tridimensional parecería que el objeto ideal debiera ser una esfera, pero este objeto tiene el inconveniente de que su mínima y su máxima curvatura son iguales. Sin embargo, nuestro método busca superficies que puedan tener una mínima y una máxima curvatura, cada una con su dirección asociada. Por lo tanto no nos valdría.

Un elipsoide por el contrario tiene diferentes curvaturas máxima y mínima, pero no es invariante a rotaciones. Le ocurre el mismo problema que a los paraboloides. Las curvaturas varían a lo largo de la superficie de distinta manera según su orientación. Además, sus curvaturas tienen siempre el mismo signo, y nuestro método puede encontrar superficies donde las curvaturas máxima y mínima tengan signo diferente. Mejor sería entonces un hiperboloide, pero tampoco es invariante a rotaciones.

¿Cuál sería entonces el objeto más conveniente para usar? Buscamos una superficie de segundo grado que pueda tener diferentes curvatura máxima y mínima, incluso de distinto signo, y que la variación de dichas curvaturas a lo largo de la superficie no dependa de su orientación. La respuesta es **un trozo de toroide**. Recordemos que la expresión de un toroide centrado en el origen, descansando sobre el plano xy, donde R y r son los radios mayor y menor respectivamente, tal y como se ve en la figura 10.24, es

$$\left(R - \sqrt{x^2 + y^2}\right)^2 + z^2 = r^2 \tag{10.8}$$

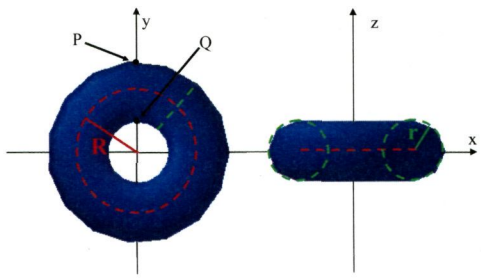


Figura 10.24: Toroide centrado en el origen sobre el plano xy

Sea el punto $P = (0, R + r, 0)$ marcado en la figura, y consideremos un diferencial de superficie centrado en dicho punto. Justo sobre ese punto existe una curvatura máxima igual a $1/r$ en la dirección del vector z, y una

curvatura mínima igual a $1/(R+r)$ en la dirección del vector x. Además, tanto la curvatura mínima como la máxima son constantes si nos movemos a lo largo de la superficie siguiendo ambas direcciones. Y es este hecho el que hace de esta figura la ideal para aproximar cualquier tipo de superficie.

Por lo tanto, sea una superficie arbitraria que pasa por un punto (x_0, y_0, z_0), cuyo vector normal en dicho punto es N, y cuyas curvaturas máxima y mínima son $k_{\max}$ y $k_{\min}$, las cuales vienen dadas a lo largo de las direcciones $v_{\max}$ y $v_{\min}$ respectivamente. Supongamos inicialmente que ambas curvaturas tienen el mismo signo. Entonces podemos afirmar que la superficie de segundo grado que mejor se aproxima es un toroide cuyos radios son

$$\begin{aligned} r &= \frac{1}{k_{\max}} \\ R &= \frac{1}{k_{\min}} - \frac{1}{k_{\max}} \end{aligned}$$

Dicho toroide, suponiendo que inicialmente se encuentra centrado en el origen y paralelo al plano xy (como en la figura 10.24), habrá de ser trasladado y rotado de forma que el punto $P = (0, R+r, 0)$ coincida con el punto (x_0, y_0, z_0), el eje y coincida con la normal N, y los ejes x y z coincidan con las direcciones $v_{\min}$ y $v_{\max}$ respectivamente. De esta forma, la aproximación obtenida será invariante a la orientación.

En el caso en que las curvaturas posean distinto signo, entonces procederemos de la misma manera pero tomando el punto $Q = (0, R-r, 0)$ de la figura 10.24. En dicho punto existe una curvatura máxima igual a $1/r$ en la dirección del vector z, y una curvatura mínima igual a $-1/(R-r)$ en la dirección del vector x. Por tanto, el toroide que buscamos es aquél cuyos radios son

$$\begin{aligned} r &= \frac{1}{k_{\max}} \\ R &= \frac{1}{k_{\max}} - \frac{1}{k_{\min}} \end{aligned}$$

Finalmente se trasladaría y rotaría de forma equivalmente a como hicimos antes.

Casos particulares Es fácil observar como para el caso en el que las curvaturas sean iguales resulta $R = 0$, y por tanto la expresión obtenida es

$$x^2 + y^2 + z^2 = r^2$$

que coincide con una esfera de radio r.

También puede verse que si una de las curvaturas es cero resulta $R = \infty$, con lo cual alrededor del punto P sabemos que x es despreciable frente a y y por tanto la expresión del toroide se asemeja a

$$(y - R)^2 + z^2 \approx r^2$$

que coincide con un cilindro de radio r orientado a lo largo del eje x.

Finalmente, si ambas curvaturas son cero, alrededor del punto P tanto x como z son despreciables frente al valor de y, y la expresión queda

$$y = R + r$$

que es la ecuación de un plano.

Complejidad de la expresión del toroide El problema de trabajar con la expresión de un toroide es que resulta muy complicado evaluar la integral doble para calcular el volumen de toroide interior a cada voxel. Téngase en cuenta que la ecuación 10.8 que representa un toroide centrado en el origen paralelo al plano xy, expresada como función explícita de x y z, queda de la siguiente manera

$$y(x,z) = \sqrt{\left(R \pm \sqrt{r^2 - z^2}\right)^2 - x^2}$$

donde el signo $\pm$ de la raiz depende de si nos referimos a la superficie vecina del punto P o del punto Q (ver figura 10.24). Dicha expresión ya es complicada de integrar sobre un recinto rectangular (la base del voxel). Pero es que después de realizar la traslación y la rotación para hacer coincidir las primeras derivadas del toroide con las de la superficie que se quiere estimar, la expresión es bastante más complicada. Por lo tanto, vamos a escoger otro objeto geométrico que, aún no siendo tan idóneo como es el toroide, sí es más sencillo de integrar.

10.4.3 Aproximación por paraboloides rotados

Hay que recordar que nuestro método detector trata siempre de encontrar el paraboloide que mejor aproxima a la superficie del contorno. Este paraboloide siempre viene dado por una expresión del tipo

$$y = a + bx + cz + dx^2 + fxz + gz^2$$

Este paraboloide es siempre de eje vertical, ya que no existe ninguno de los términos cruzados xy ni yz. Esto significa que en los lugares de pendiente horizontal la aproximación es bastante acertada, pero en las zonas cercanas a las bisectrices de los planos principales, el error cometido empieza a ser considerable. Sin embargo, si estimásemos un paraboloide rotado, es decir, un paraboloide cuyo eje central coincida con la dirección del vector normal a la superficie sobre el punto donde queremos aproximar, el resultado sería mucho más exacto. Veamos primero un ejemplo en 2 dimensiones y luego pasaremos a desarrollar el caso general primero en 2D y finalmente en 3D.

Ejemplo 2D

Tomemos una circunferencia de radio 1, centrada en el punto $(0,1)$. La parábola vertical que mejor aproxima a la curva en el punto $(0,0)$ es

$$y = \frac{1}{2}x^2$$

En la figura 10.25a se aprecia que la diferencia entre el arco de circunferencia y la parábola es inapreciable cerca del origen.

Ahora tomemos la misma circunferencia de radio 1 pero centrada en el punto $(-1/\sqrt{2}, 1/\sqrt{2})$. La parábola vertical que mejor aproxima a la curva en el punto $(0,0)$ viene dada por la ecuación $y = bx + cx^2$, donde la pendiente ha de ser 1 en el origen ($b = 1$) y la curvatura también 1

$$K = \frac{2c}{(1+b)^{3/2}} = 1 \Rightarrow c = \sqrt{2}$$

Por lo tanto, la expresión de la parábola vertical es

$$y = x + \sqrt{2}x^2$$

En la figura 10.25b puede verse la diferencia existente entre ambas curvas.

Sin embargo, la parábola rotada es diferente. Considerando el sistema $\{u, v\}$ como las bisectrices del sistema $\{x, y\}$, tendríamos que las expresiones del nuevo sistema son

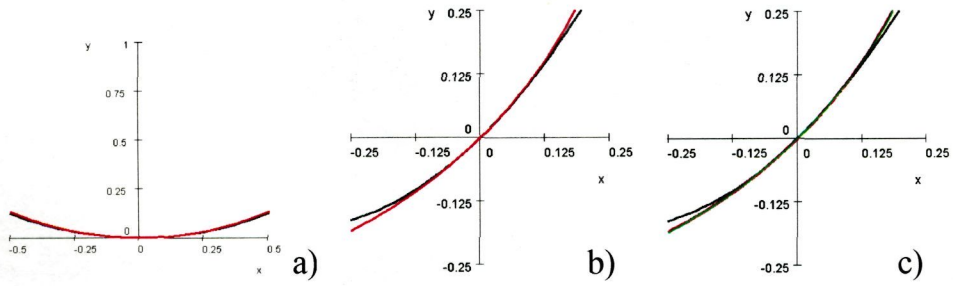


Figura 10.25: Aproximación de una circunferencia (color rojo) usando una parábola vertical (color negro) y una parábola rotada (color verde)

$$\begin{aligned} u &= \frac{1}{2}\sqrt{2}x + \frac{1}{2}\sqrt{2}y \\ v &= -\frac{1}{2}\sqrt{2}x + +\frac{1}{2}\sqrt{2}y \end{aligned}$$

La expresión de la parábola rotada en el sistema $\{u, v\}$ es

$$v = \frac{1}{2}u^2$$

con lo cual en el sistema $\{x, y\}$ pasa a ser

$$\frac{1}{2}xy + \frac{1}{4}x^2 + \frac{1}{4}y^2 + \frac{1}{\sqrt{2}}x - \frac{1}{\sqrt{2}}y = 0$$

En la figura 10.25c se muestra la gráfica de la parábola rotada, y se puede apreciar cómo ésta es mucho más parecida a la curva original.

Caso general 2D

Dada una curva $f(x)$, nuestro método detector encuentra que en el punto $(0, a)$ la parábola estimada es del tipo

$$y = a + bx + cx^2$$

Deduzcamos cuál es la parábola rotada que comparte pendiente y curvatura con la parábola vertical en dicho punto. Dicha parábola rotada siempre será del tipo

$$v = ku^2$$

donde el origen del sistema $\{u, v\}$ se encuentra en el punto $(0, a)$, y el eje v coincide con el vector normal N a la parábola en dicho punto, cuyas compontentes son $N = [b, -1]$.

Por lo tanto, los ejes del sistema $\{u, v\}$ vienen dados por la expresión

$$\begin{aligned} v &= \frac{1}{M}\begin{bmatrix} -b & 1 \end{bmatrix} \\ u &= \frac{1}{M}\begin{bmatrix} 1 & b \end{bmatrix} \end{aligned}$$

donde $M = \sqrt{1+b^2}$. Las ecuaciones para cambiar de sistema son:

$$\begin{aligned} u &= u_x x + u_y (y-a) = \frac{1}{M}\left[x + b(y-a)\right] \\ v &= v_x x + v_y (y-a) = \frac{1}{M}\left[-bx + y - a\right] \end{aligned}$$

Por tanto, la expresión de la parábola rotada queda:

$$v_x x + v_y (y-a) - k\left(u_x x + u_y (y-a)\right)^2 = 0$$

Nótese que la curvatura de la parábola rotada en $(0,0)$ es $K = 2k$, mientras que la de la parábola vertical en $(0,a)$ es $K = 2c/\left(1+b^2\right)^{3/2}$. Por lo tanto se cumple que

$$k = \frac{c}{(1+b^2)^{3/2}} = \frac{c}{M^3}$$

Sustituyendo todo obtenemos la expresión final de la parábola rotada en el sistema $\{x, y\}$:

$$\left(2abc - M^4 b\right) x + \left(2ab^2 c + M^4\right) y - (2bc)\, xy - cx^2 - b^2 c y^2 - \left(M^4 a + a^2 b^2 c\right) = 0$$

Caso general 3D

Dada una superficie $f(x,z)$, nuestro método detector encuentra que en el punto $(0,a,0)$ el paraboloide estimado es del tipo

$$y = a + bx + cz + dx^2 + fxz + gz^2$$

Deduzcamos cuál es el paraboloide rotado que comparte normal y curvaturas con el paraboloide vertical en dicho punto. Dicho paraboloide rotado siempre será del tipo

$$v = k_1 u^2 + k_2 w^2$$

donde el origen del sistema $\{u, v, w\}$ se encuentra en el punto $(0,a,0)$, el vector v coincide con el vector normal al paraboloide en dicho punto, $[b,-1,c]$, el vector u coincide con el vector de máxima curvatura, y el w con el de mínima curvatura.

Las curvaturas del paraboloide rotado en el punto $(0,0,0)$ son $K_{\max} = 2k_1$ y $K_{\min} = 2k_2$

Las ecuaciones para cambiar de sistema son

$$\begin{aligned} u &= u_1 x + u_2 (y-a) + u_3 z \\ v &= v_1 x + v_2 (y-a) + v_3 z \\ w &= w_1 x + w_2 (y-a) + w_3 z \end{aligned}$$

Por tanto el paraboloide rotado queda

$$v_1 x + v_2 (y-a) + v_3 z = \frac{1}{2} K_{\max} \left(u_1 x + u_2 (y-a) + u_3 z\right)^2 + \frac{1}{2} K_{\min} \left(w_1 x + w_2 (y-a) + w_3 z\right)^2$$

Este resultado se resume en el siguiente lema:

Lema 10.1 *Sea el paraboloide vertical dado por la ecuación* $y = a+bx+cz+dx^2+fxz+gz^2$. *El paraboloide rotado que comparte posición, orientación y curvaturas en el punto* $(0, a, 0)$ *viene dado por la expresión*

$$A + Bx + Cy + Dz + Exy + Fxz + Gyz + Hx^2 + Iy^2 + Jz^2 = 0$$

donde

$$\begin{aligned}
A &= -a\left(v_2 + \frac{1}{2}aK_{max}u_2^2 + \frac{1}{2}aK_{min}w_2^2\right)\\
B &= v_1 + aK_{\max}u_1u_2 + aK_{\min}w_1w_2\\
C &= v_2 + aK_{\max}u_2^2 + aK_{\min}w_2^2\\
D &= v_3 + aK_{\max}u_2u_3 + aK_{\min}w_2w_3\\
E &= -\left(K_{\max}u_1u_2 + K_{\min}w_1w_2\right)\\
F &= -\left(K_{\max}u_1u_3 + K_{\min}w_1w_3\right)\\
G &= -\left(K_{\max}u_2u_3 + K_{\min}w_2w_3\right)\\
H &= -\frac{1}{2}\left(K_{\max}u_1^2 + K_{\min}w_1^2\right)\\
I &= -\frac{1}{2}\left(K_{\max}u_2^2 + K_{\min}w_2^2\right)\\
J &= -\frac{1}{2}\left(K_{\max}u_3^2 + K_{\min}w_3^2\right)
\end{aligned}$$

y donde $K_{\max}$ *y* $K_{\min}$ *son las curvaturas máximas y mínimas del paraboloide vertical en el punto* $(0, a, 0)$, $v = [v_1, v_2, v_3]$ *es el vector normal al paraboloide en dicho punto, y* $u = [u_1, u_2, u_3]$ *y* $w = [w_1, w_2, w_3]$ *son las direcciones de máxima y mínima curvaturas respectivamente en dicho punto. Todos estos valores se deducen con el lema 8.1.*

10.4.4 Detección precisa del paraboloide rotado

Recordemos que al principio de la sección comentamos los dos errores principales cometidos por nuestro método cuando es aplicado a contornos con alta curvatura. Con el lema anterior hemos solucionado el segundo de ellos, es decir, el concerniente a la generación de las subimágenes sintéticas en la vecindad de cada voxel borde. Simplemente calculando la integral de los paraboloides rotados en lugar de los verticales mejoraremos la precisión de las imágenes generadas en cada iteración (en el anexo C.3 se muestra el método de integración para los paraboloides rotados).

Sin embargo, el primer problema aún sigue sin resolverse. Atendiendo a la figura 10.21b, por muy precisas que sean las subimágenes sintéticas generadas, ya el resultado obtenido por el método detector contiene un error que nos hará ir deformando el objeto. Esto es debido a que el paraboloide vertical obtenido por nuestro método detector no es realmente el que comparte orientación y curvatura con la superficie del contorno, sino el que mejor se aproxima a los valores de las intensidades de los voxels vecinos, asumiendo que el contorno sigue una expresión de segundo grado.

Por tanto necesitamos lograr también una precisión mayor en la estimación de los parámetros del contorno (posición, orientación y curvaturas). Para ello, procederemos de forma idéntica a como se hizo en la sección 6.2.1 para el caso bidimensional. Pero antes necesitamos desarrollar un nuevo lema, justo el inverso del lema 10.1.

Estimación del paraboloide vertical

Sea el paraboloide rotado dado por la expresión $A + Bx + Cy + Dz + Exy + Fxz + Gyz + Hx^2 + Iy^2 + Jz^2 = 0$, y deseamos calcular el paraboloide vertical que comparte posición, normal y curvaturas en el punto (x_0, y_0, z_0).

Para ello haremos uso del desarrollo de Taylor de grado 2 para una función de dos variables, $f(x,z)$, dado por la expresión

$$y \simeq y_0 + f_x(x-x_0) + f_z(z-z_0) + \frac{1}{2}f_{xx}(x-x_0)^2 + f_{xz}(x-x_0)(z-z_0) + \frac{1}{2}f_{zz}(z-z_0)^2 \tag{10.9}$$

donde todas las derivadas parciales están evaluadas en el punto (x_0, y_0, z_0).

Como la expresión del paraboloide rotado puede verse de la forma $P(x,y,z) = 0$, podemos obtener las derivadas parciales de la variable y respecto de las otras dos a partir de la derivación implícita, de la siguiente manera:

$$\begin{aligned}
f_x(x,y,z) &= -\frac{P_x}{P_y} \\
f_z(x,y,z) &= -\frac{P_z}{P_y} \\
f_{xx}(x,y,z) &= -\frac{P_{xx} + 2P_{xy}f_x + P_{yy}f_x^2}{P_y} \\
f_{xz}(x,y,z) &= -\frac{P_{xz} + P_{xy}f_z + P_{yz}f_x + P_{yy}f_xf_z}{P_y} \\
f_{zz}(x,y,z) &= -\frac{P_{zz} + 2P_{yz}f_z + P_{yy}f_z^2}{P_y}
\end{aligned}$$

donde P_x, P_y y P_z son las derivadas parciales de $P(x,y,z)$. Evaluando estas derivadas en el punto (x_0, y_0, z_0) y sustituyéndolas en la expresión 10.9 obtendremos el paraboloide vertical buscado. Este resultado se resume en el siguiente lema:

Lema 10.2 *Sea el paraboloide rotado dado por la ecuación* $A+Bx+Cy+Dz+Exy+Fxz+Gyz+Hx^2+Iy^2+Jz^2=0$. *El paraboloide vertical que comparte posición, orientación y curvaturas en el punto* (x_0, y_0, z_0) *viene dado por la expresión*

$$y = a + bx + cz + dx^2 + fxz + gz^2$$

donde

$$\begin{aligned}
a &= y_0 - x_0f_x - z_0f_z + x_0z_0f_{xz} + \frac{1}{2}x_0^2f_{xx} + \frac{1}{2}z_0^2f_{zz} \\
b &= f_x - z_0f_{xz} - x_0f_{xx} \\
c &= f_z - x_0f_{xz} - z_0f_{zz} \\
d &= \frac{1}{2}f_{xx} \\
f &= f_{xz} \\
g &= \frac{1}{2}f_{zz}
\end{aligned}$$

y donde

$$\begin{aligned} f_x &= -\frac{1}{P_y}(B + Fz_0 + 2Hx_0 + y_0E) \\ f_z &= -\frac{1}{P_y}(D + Fx_0 + Gy_0 + 2Jz_0) \\ f_{xx} &= -\frac{1}{P_y}\left(2H + 2Ef_x + 2If_x^2\right) \\ f_{xz} &= -\frac{1}{P_y}(F + Ef_z + Gf_x + 2If_xf_z) \\ f_{zz} &= -\frac{1}{P_y}\left(2J + 2Gf_z + 2If_z^2\right) \end{aligned}$$

siendo

$$P_y = C + Gz_0 + 2y_0I + x_0E$$

Método numérico

Vayamos ahora con el método numérico propuesto para detectar el paraboloide rotado buscado. La idea es la siguiente: asumamos que la superficie del contorno sigue la expresión de un paraboloide rotado, cuya expresión es:

$$A + Bx + Cy + Dz + Exy + Fxz + Gyz + Hx^2 + Iy^2 + Jz^2 = 0$$

y cuyos coeficientes desconocemos a priori. Lo que sí podemos conocer en primer lugar es cuál es la expresión del paraboloide vertical, P_E, obtenido por nuestro método detector, a partir del lema 9.3, cuya expresión llamaremos

$$P_E(x,z) = a_E + b_Ex + c_Ez + d_Ex^2 + f_Exz + g_Ez^2$$

Sin embargo, el paraboloide vertical ideal, P_I, que desearíamos obtener, suponiendo conocida la expresión exacta del contorno, sería el obtenido con el lema 10.2 que acabamos de ver, dado por la expresión

$$P_I(x,z) = a_I + b_Ix + c_Iz + d_Ix^2 + f_Ixz + g_Iz^2$$

Haciendo las restas de cada pareja de coeficientes

$$\begin{aligned} \varepsilon_a &= a_E - a_I \\ \varepsilon_b &= b_E - b_I \\ \varepsilon_c &= c_E - c_I \\ \varepsilon_d &= d_E - d_I \\ \varepsilon_f &= f_E - f_I \\ \varepsilon_g &= g_E - g_I \end{aligned}$$

obtenemos seis factores de corrección que, restados a los coeficientes obtenidos por nuestro método detector, nos darían los coeficientes del paraboloide ideal buscado. Pero el problema es que el paraboloide rotado que representa al contorno es desconocido a priori. Sin embargo, tenemos una buena aproximación de él, dado por el paraboloide rotado que comparte posición, normal y curvaturas en el punto $(0, a, 0)$ con el paraboloide obtenido por nuestro método.

Por tanto, puede plantearse el algoritmo siguiente:

1. Dado un voxel por el que pasa un contorno desconocido, aplicamos nuestro método detector para obtener un paraboloide vertical, P^0, a partir del cual se obtiene la estimación de la posición, normal y curvaturas del contorno (lema 9.3).
2. Con dichos valores deducimos el paraboloide rotado, R^0, que comparte posición, normal y curvaturas con P^0 (lema 10.1). Este paraboloide R^0 es una buena aproximación al contorno.
3. A partir de R^0 se calculan dos paraboloides verticales: el que estimaría nuestro método, P_E^0, aplicando el lema 9.3, y el que comparte normal y curvaturas, P_I^0, aplicando el lema 10.2. Usando estos dos paraboloides se deducen los factores de corrección ε^0.
4. Usando los factores ε^0 podemos corregir el paraboloide vertical inicial, P^0, para obtener un nuevo paraboloide vertical corregido, P^1, más exacto que el anterior.
5. A partir de P^1 deducimos el paraboloide rotado, R^1, que comparte posición, normal y curvaturas (lema 10.1). Este paraboloide R^1 será una mejor aproximación del contorno.

De hecho, los pasos 3 al 5 de este esquema pueden repetirse sucesivamente hasta converger. Es decir, en cada iteración n, partimos de un paraboloide rotado R^n con el cual calculamos los factores de corrección ε^n para corregir el paraboloide vertical inicial P^0 y obtener P^{n+1} del cual deducimos un nuevo paraboloide rotado R^{n+1}, que será una mejor aproximación a la superficie del contorno. Esto puede expresarse como

$$P^{n+1} = P^0 + P_I^n - P_E^n$$

Teniendo en cuenta que $P_I^n = P^n$, la expresión recursiva para los nuevos paraboloides queda

$$P^{n+1} = P^n + \left(P^0 - P_E^n\right)$$

El algoritmo final quedaría como sigue:

Funcion EstimarParaboloideRotado (P, R)

```
P0 = P
Repetir hasta convergencia
   Deducir el paraboloide rotado R a partir de P (lema 10.1)
   Crear una subimagen F' de 5x5x5 centrada en el voxel a partir de R (anexo C.3)
   Suavizar F' para obtener G'
   Calcular el paraboloide vertical estimado Pe a partir de la imagen G' (lema 9.3)
   P += P0 - Pe
Fin Repetir
```

El algoritmo parte del paraboloide vertical obtenido por el método detector, P, y en cada iteración se va obteniendo un nuevo paraboloide vertical cuya posición, dirección normal y curvaturas, medidos sobre la vertical central del voxel, se va acercando a los verdaderos valores de la superficie del contorno.

Otro detalle que hay que mencionar es la dimensión de las ventanas usadas. Ya que el objetivo es poder detectar altas curvaturas, debemos usar ventanas centradas en el voxel del menor tamaño posible, ya que cuanto mayor sea la ventana, más promediada será la información que obtengamos. Por lo tanto, una ventana de $3x3x3$ sobre la imagen original ($5x5x5$ en la imagen suavizada) será la mejor elección.

Por otro lado hay que recordar que originalmente se usaban ventanas más alargadas para poder garantizar que la superficie del contorno no tocara nunca el suelo ni el techo de la ventana, y poder trabajar así con integrales sencillas para calcular los volúmenes de forma exacta en cada columna de voxels. Sin embargo, como ahora tenemos un método iterativo para obtener la expresión de la superficie, no es relevante que la superficie toque el suelo de la ventana y el resultado de la integral sea del todo exacto, ya que finalmente el método numérico convergerá a la superficie.

10.4.5 Algoritmo final

Este esquema de detección de curvaturas altas a partir de subimágenes $3x3x3$ debe también combinarse con el esquema propuesto para la detección de contornos muy cercanos (método de límites variables de la sección 9.4). La respuesta es sencilla: centrándonos de nuevo en una imagen angiográfica, puede ocurrir que una vena de muy poco grosor tenga además una curvatura alta, con lo cual incluso en una ventana $3x3x3$ ($5x5x5$ antes de suavizar) pueden existir más de un contorno en el interior de la subimagen. Para poder abarcar también con precisión estos casos, debemos realizar conjuntamente ambos procesos.

La solución propuesta es la siguiente: para la estimación de A y B haremos igual que en el algoritmo previo, es decir, buscando en cada columna donde se alcanza un mínimo de la parcial, y usando esos voxels para la estimación. Además dichos límites nos servirán para decidir si existe algún otro contorno muy cercano al que estamos tratando, tanto por arriba como por abajo. En este caso generaremos una subimagen sintética donde aplicar el detector. Finalmente, aplicaremos la detección propuesta en esta sección a partir de una subimagen $3x3x3$. Por tanto, el algoritmo final de restauración quedaría como sigue:

Funcion RestaurarImagen3D_optimizado (F0, F1, delta)

```
Crear una imagen de contadores C e inicializarla a 0
Crear una imagen de intensidades I e inicializarla a 0
Suavizar F con una mascara H para obtener la imagen G
Calcular las parciales Gx, Gy, Gz

// bucle principal
Para todos los voxels borde (i,j,k) de la imagen G
      P = DetectarContornos3D (i, j, k, orientacion, F, G)
      EstimarParaboloideRotado (P, R)
      ActualizarImagenes (C, I, i, j, k, orientacion, R)
Fin Para

// generacion de la imagen restaurada
Crear una imagen F1 = G
Para todos los voxels (i,j,k) de la imagen G
   Si C(i,j,k) > 0 Entonces F1(i,j,k) = I(i,j,k) / C(i,j,k)
Fin Para
```

10.4.6 Ejemplos sintéticos

Detección

El primer ejemplo que podemos poner es la misma imagen de la esfera de radio 10 de la figura 10.21, donde el error máximo cometido en la estimación de la dirección del vector normal era cercano a 4 grados. Usando paraboloides rotados el error máximo es mucho menor (cercano a 0.15 grados) como puede verse en la figura 10.26, donde también se aplica el método a esferas de radio menor (7 y 5 voxels). El error máximo en esta última es inferior a 2 grados.

En la figura 10.27 se muestra la detección en distintos toroides. Centremos primero nuestra atención en la fila superior. En ella se muestra la detección de los valores y direcciones de curvatura en un toroide de radios 10 y 3. Puede verse como los resultados son bastante precisos. El primer toroide de la izquierda muestra el error cometido en la estimación del valor de curvatura máxima en cada voxel de la superficie. El color azul indica que no hay error (es decir, la curvatura máxima es 1/3, o lo que es lo mismo, el radio de curvatura es 3).

Vemos que en la mayoría de los voxels el radio de curvatura estimado es correcto. El segundo toroide muestra las direcciones de curvatura máxima, las cuales deben ser tangentes al perímetro de la sección.

El tercer toroide muestra el valor concreto de la curvatura mínima, la cual debería oscilar entre $1/13$ en los voxels exteriores y $-1/7$ en los voxels interiores donde se encuentra el agujero, tal y como se ve en la figura. Finalmente el cuarto toroide muestra las direcciones de mínima curvatura, las cuales apuntan a la dirección del toroide.

En las filas inferiores se muestran otros dos toroides con su radio menor aún más pequeño (2 y 1 voxel), y los resultados ya no son tan precisos. Téngase en cuenta que en cada iteración del método numérico propuesto se realiza un cálculo de volúmenes en el interior de cada voxel, y a su vez ese cálculo se realiza mediante un esquema numérico de integración. Además, cuando la curvatura es muy elevada (como en la fila inferior) también existe un error entre aproximar la superficie de un toroide por la de un paraboloide rotado. Todo ello hace que se pierda precisión. Sin embargo, visualmente, la estimación de los contornos es aceptable.

Restauración

En la figura 10.28 se muestra la imagen sintética de una estenosis[3], en donde el grosor de la parte más ancha es de 10 voxels, y la estrechez del centro de la imagen es de 5 voxels, y a la cual se le ha añadido ruido. La imagen a su derecha muestra el resultado de la restauración. Finalmente se muestra también la reconstrucción obtenida en cada voxel borde. El color en cada voxel muestra el grosor estimado. Para ello, teniendo en cuenta que la inversa de la curvatura máxima representa el radio de curvatura del vaso, el grosor puede expresarse como $2/k_{\max}$. Puede apreciarse cómo la mayoría de los voxels borde de la zona ancha del vaso tienen un grosor próximo a 10, y la zona de la estrechez se acerca a las 5 unidades.

En la figura 10.29a se muestra el resultado de la restauración a una imagen sintética, donde se muestra un vaso cuyo grosor oscila entre 6 y 12 voxels, a la cual se le ha añadido ruido. El color en la reconstrucción muestra el grosor estimado en cada voxel de la superficie. Puede observarse cómo salvo algunos voxels aislados con grosor elevado (color rojo) el resultado es bastante satisfactorio (azul en las zonas más estrechas y verde en las más anchas).

Hay que tener en cuenta que el error en los voxels que aparecen en color rojo no es demasiado grande, ya que la escala del grosor no es lineal. Por ejemplo, en las zonas donde el grosor es 12 voxels, el valor exacto de la curvatura máxima debería ser $1/6 \approx 0.167$. Sin embargo, el color rojo de la figura indica que el grosor es cercano a 24 voxels, es decir, la curvatura máxima estimada fue de $1/12 \approx 0.083$. Realmente el error ha sido entonces aproximadamente de 0.08 unidades, lo cual es un error muy pequeño. Téngase en cuenta que para valores de curvatura bajos, existe muy poca diferencia entre un diferencial de superficie y otro.

En la figura 10.29b se muestra el mismo proceso, pero esta vez sobre la imagen de un vaso más alargado, cuyo grosor oscila entre 4 y 8 voxels. De nuevo puede apreciarse como, a excepción de algunos voxels aislados en color rojo, el grosor vuelve a ser satisfactorio (azul en las zonas más estrechas y verde en las más anchas).

10.5 Ejemplos con imágenes reales

Objetos artificiales La figura 10.30 muestra el scanner tridimensional de una pieza mecánica. A la derecha de la imagen se muestra una superficie de nivel para apreciar mejor su geometría. La pieza está formada por una cruz y dos arcos conectados entre sí. La imagen ha sido restaurada con nuestro método, y en la figura derecha se aprecian los contornos estimados sobre la imagen restaurada. El color de cada voxel muestra el grosor detectado en cada uno. Podemos apreciar cómo en las zonas de las intersecciones el grosor estimado es mayor que sobre la zona de los cilindros.

La figura 10.31 muestra el scanner de una pieza de un motor. Igualmente se muestra la isosuperficie, y a su derecha los contornos detectados sobre la imagen restaurada. El color sigue representando el grosor estimado.

[3] Se conoce por estenosis a la zona de un vaso sanguíneo donde el grosor se reduce bruscamente.

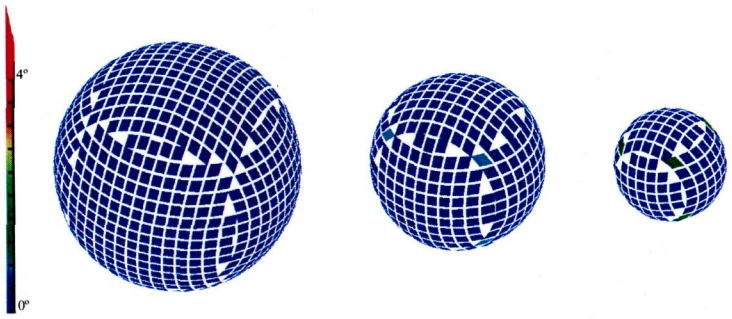


Figura 10.26: Error cometido en la estimación de la dirección normal sobre una esfera, usando el método de detección de paraboloides rotados. De izquierda a derecha, el radio es 10, 7 y 5 voxels.

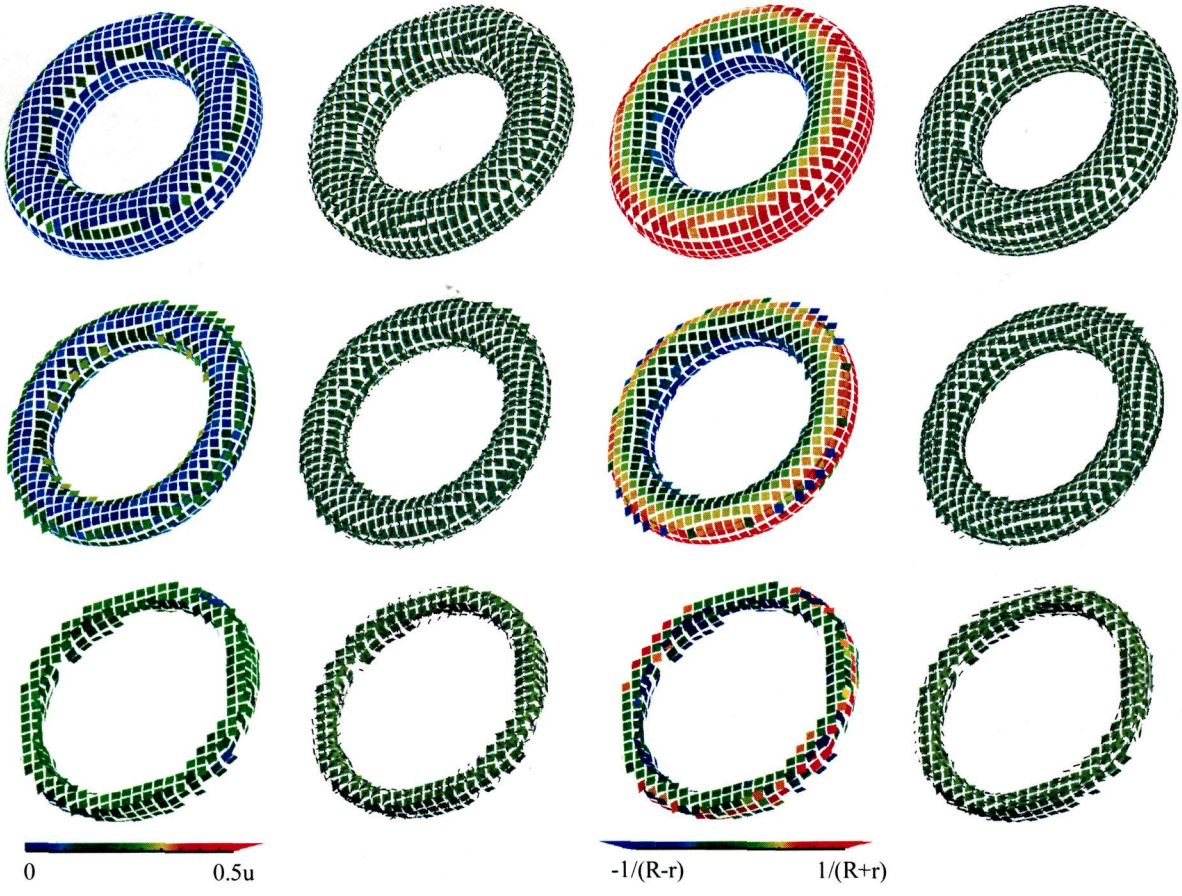


Figura 10.27: Detección de las curvaturas en un toroide de radio mayor 10 y radio menor (de arriba a abajo) 3, 2 y 1 voxel. De izquierda a derecha: error cometido en el valor de curvatura máxima; dirección de curvatura máxima; valor de curvatura mínima; dirección de curvatura mínima.

Se puede apreciar cómo las zonas con mayor curvatura tienen color azul, el codo de las toberas superiores muestra una curvatura menor (color verde) y las zonas más planas son de color rojo (curvatura muy pequeña).

Tomografía craneal La figura 10.32 muesta una tomografía (CT) de una cabeza humana. Los voxels tienen menor longitud en la dirección z que en las direcciones x e y. Los valores de intensidad pertenecen al intervalo $[0, 255]$. Podemos considerar que la imagen posee tres zonas de intensidad diferenciada: el fondo oscuro de la imagen, con valores cercanos a 10, la zona blanda de la cabeza, con valores cercanos a 80, y la zona dura perteneciente a los huesos, con valores cercanos a 180. Esto significa que los contornos de la parte blanda tendrán un salto de intensidad cercano a las 70 unidades, mientras que los contornos de los huesos tendrán un salto de intensidad cercano a las 100 unidades.

A la derecha de la imagen se muestra la estimación de los contornos sobre la imagen, una vez restaurada. El color de cada voxel indica la magnitud del salto de intensidad. La figura central representa los contornos cuyo salto de intensidad es mayor que 60, y por tanto se muestra la zona exterior de la cabeza. En la figura de la derecha se muestran los contornos con salto mayor que 90, y por tanto lo que se aprecia es la superficie del cráneo. Las zonas donde no se aprecia contorno alguno es debido a que el salto de intensidad es inferior.

Angiografías La figura 10.33 muestra una angiografía renal. Su tamaño es de $256x256x122$ voxels. Su intensidad oscila entre 0 y 1400 unidades. En la figura se muestra el resultado de la restauración y los contornos estimados en algunas zonas. Las dos figuras superiores muestran una vista general, donde el color de cada voxel indica la magnitud del salto de intensidad. En las dos figuras inferiores se muestra con más detalle algunas zonas, donde el color ahora representa el grosor estimado para la superficie en cada voxel, siendo el color azul para las zonas más estrechas y rojo para las zonas más planas.

La figura 10.34 es similar, pero aplicado sobre una angiografía craneal de $256x256x60$, cuya intensidad oscila entre 0 y 900 unidades. Hay que tener en cuenta que el objetivo buscado no es tanto la representación visual de los contornos sino la estimación precisa de los parámetros de la superficie en cada voxel borde.

Finalmente, la figura 10.35 muestra la misma imagen utilizada en la figura 10.19, pero aplicando el método de restauración propuesto para altas curvaturas. La imagen tiene una resolución de $128x128x128$ voxels, con un rango de intensidad entre 0 y 3300. La fila central de la figura muestra una isosuperficie de nivel 1000, y la estimación de contornos con el método propuesto, indicando con el color la magnitud del salto de intensidad a ambos lados del contorno (imagen central) y el grosor estimado en cada voxel borde (imagen derecha). Vemos que el resultado es bastante satisfactorio.

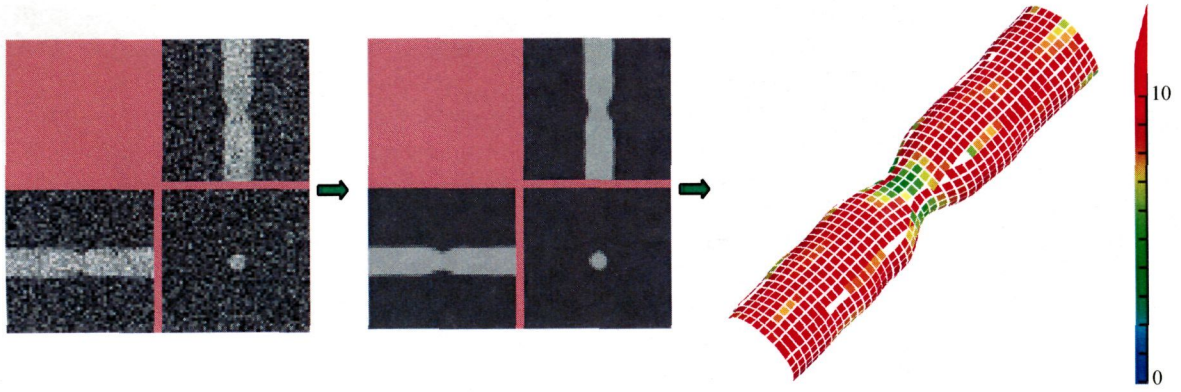


Figura 10.28: Restauración de una estenosis sintética: un vaso de grosor 10 con una estrechez de grosor 5. El color indica el grosor detectado en cada voxel

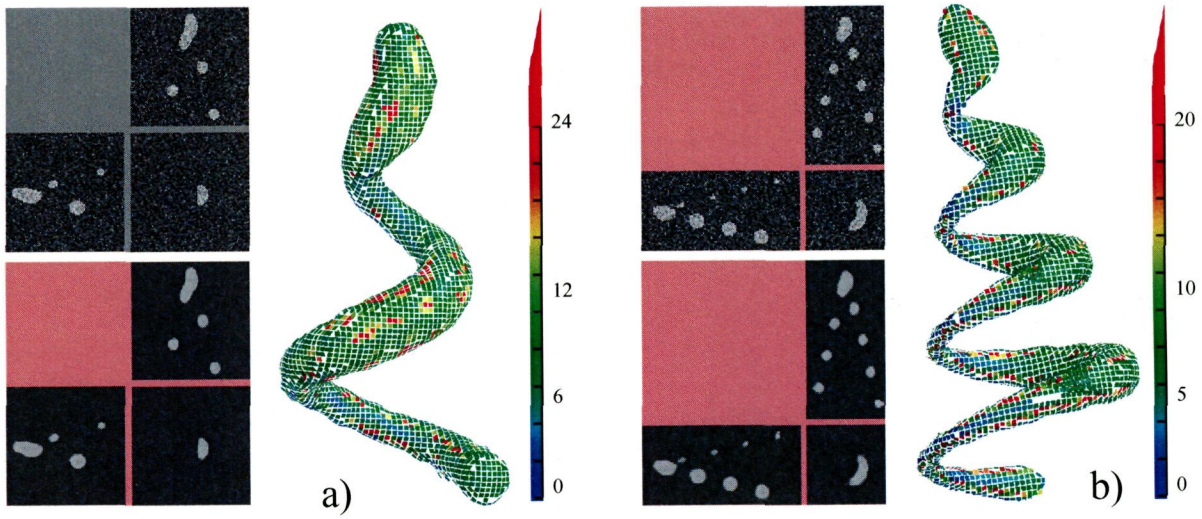


Figura 10.29: Restauración y estimación de contornos en dos imágenes sintéticas. El color en cada voxel muestra el grosor estimado en cada uno.

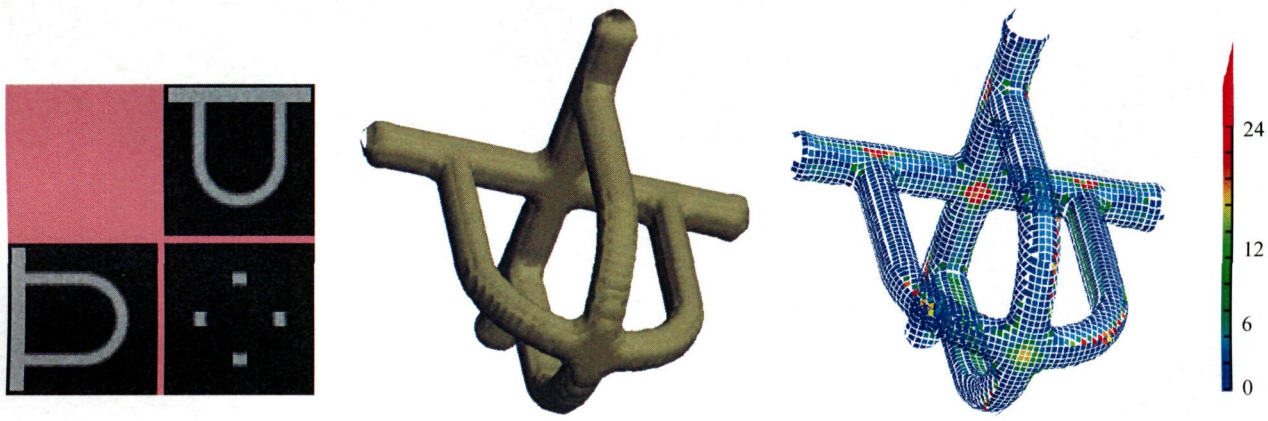


Figura 10.30: Scanner de una pieza mecánica. De izquierda a derecha: imagen original, isosuperficie, contornos estimados con el color indicando el grosor.

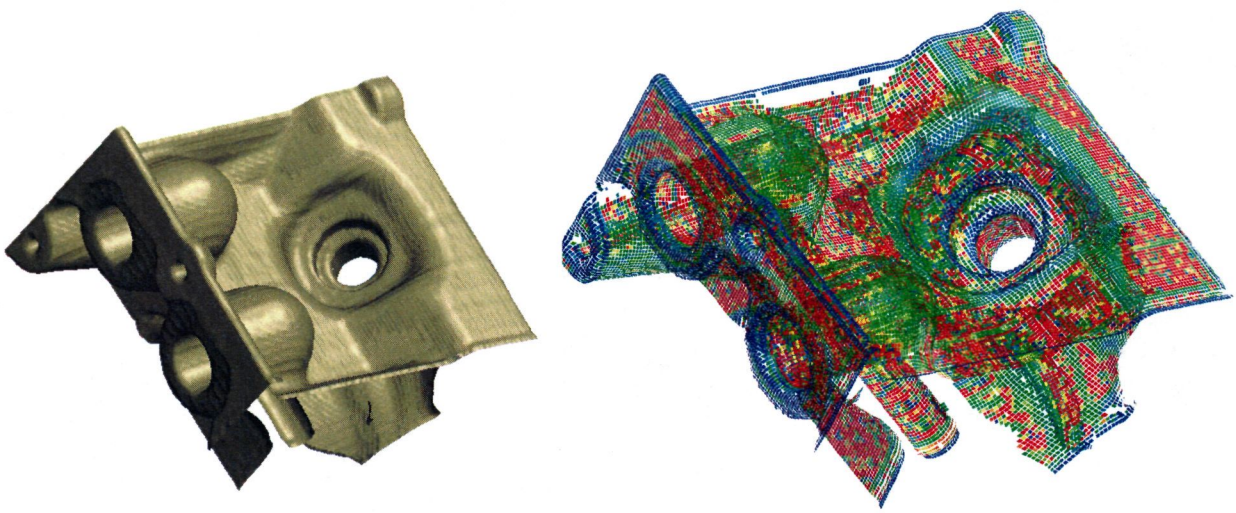


Figura 10.31: Scanner de una pieza de un motor. De izquierda a derecha: isosuperficie, contornos estimados con el color indicando el grosor (medido como la inversa del radio de curvatura máxima).

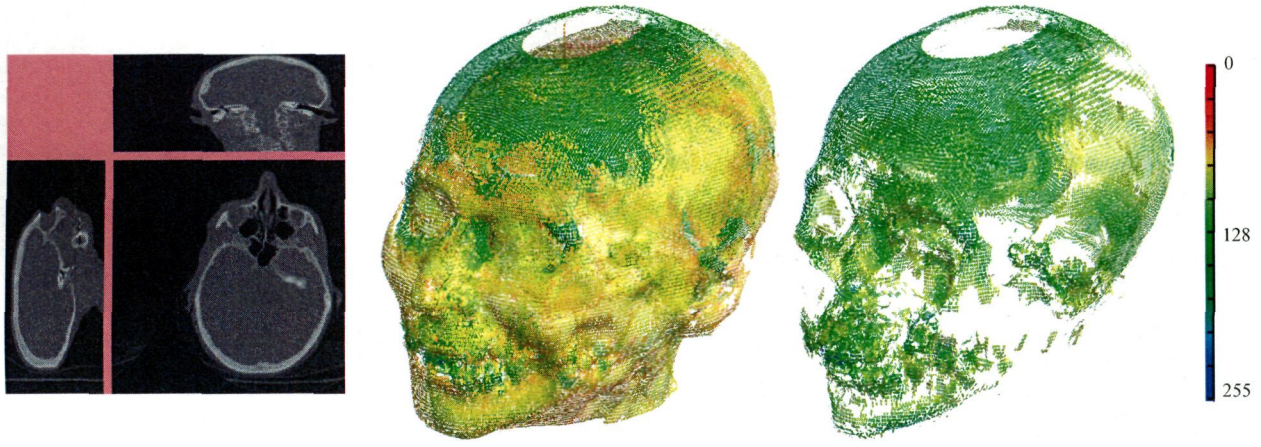


Figura 10.32: Tomografía craneal. De izquierda a derecha: imagen original; contornos estimados con salto de intensidad mayor de 60;ídem con salto mayor que 90.

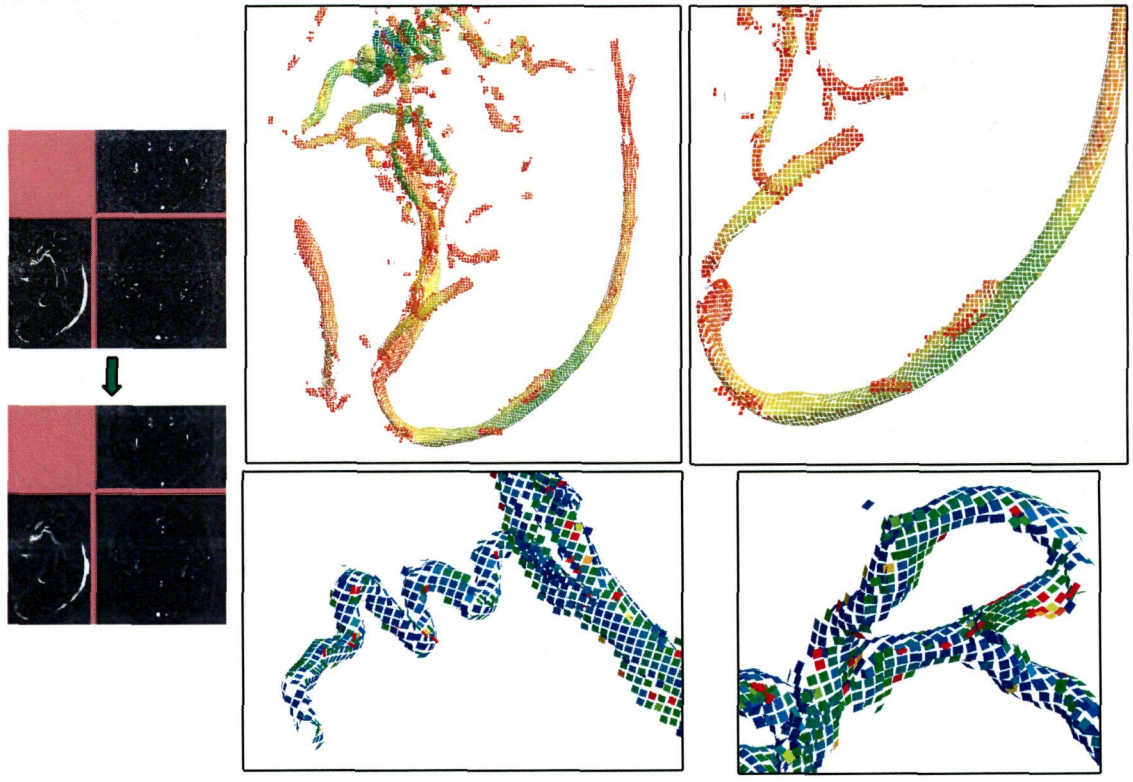


Figura 10.33: Restauración de una angiografía renal, y estimación de contornos. El color en cada voxel indica el salto de intensidad (figuras superiores) y el grosor (figuras inferiores)

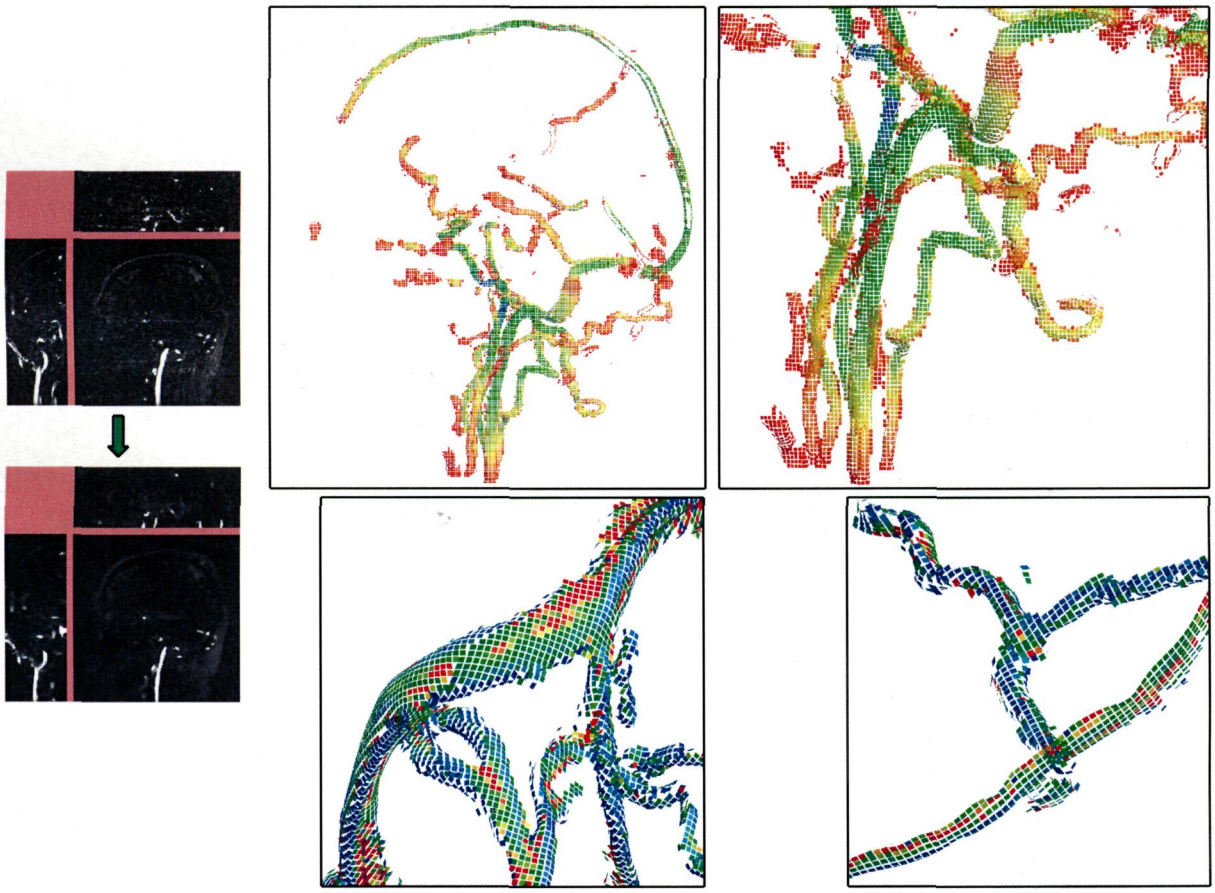


Figura 10.34: Restauración de una angiografía craneal, y estimación de contornos. El color en cada voxel indica el salto de intensidad (figuras superiores) y el grosor (figuras inferiores)

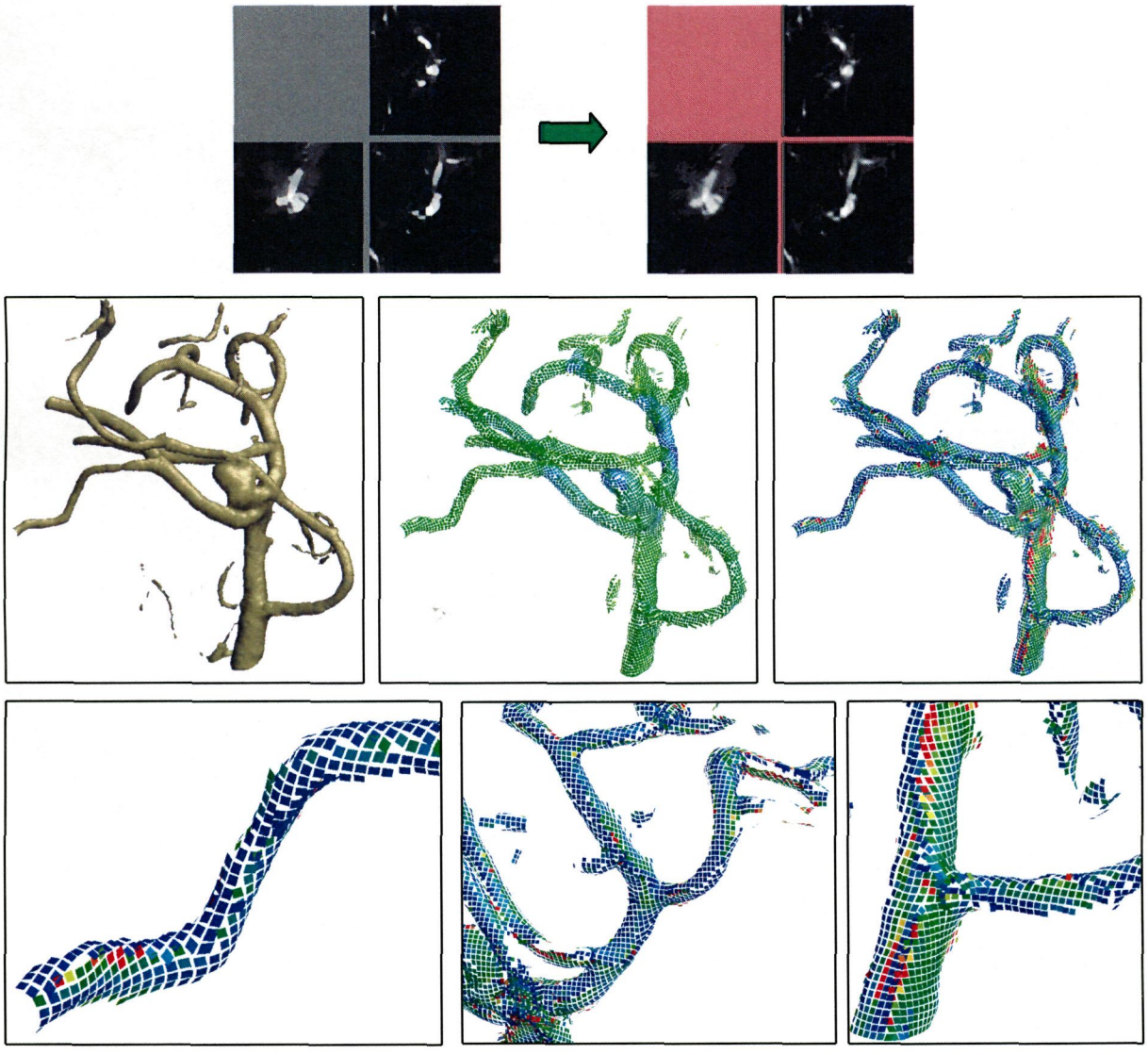


Figura 10.35: De izquierda a derecha, fila superior: imagen original y restauración; fila central: isosuperficie y estimación de contornos, donde el color en cada voxel indica el salto de intensidad (central) y el grosor (derecha); detalle de algunas zonas, con el color indicando el grosor

Parte III

Conclusiones y anexos

Capítulo 11

Conclusiones y líneas futuras

Como mencionábamos al principio de esta tesis, el objetivo de este trabajo ha sido el realizar un estudio en profundidad de cómo poder deducir los parámetros intrínsecos de un borde (orientación, curvatura, posición sub-pixel y salto de intensidad a ambos lados), partiendo de los valores de los pixels en la imagen.

Para lograr dicho objetivo hemos planteado un modelo de borde y un modelo de adquisición, tanto para imágenes 2D como 3D, los cuales se han usado para generar imágenes sintéticas ideales con las cuales trabajar. Los resultados de la investigación han conseguido obtener en este tipo de imágenes ideales todos estos parámetros de forma exacta. Lo anterior significa que, en tanto en cuanto los modelos de borde y adquisición sean lo suficientemente realistas, la precisión obtenida en imágenes reales será bastante alta, incluso en presencia de ruido.

11.1 Resultados obtenidos

Podemos clasificar separadamente los resultados y conclusiones obtenidos para imágenes bidimensionales y para imágenes tridimensionales.

11.1.1 Resultados en imágenes 2D

Centrándonos inicialmente en el caso de imágenes 2D, los principales logros obtenidos en el presente trabajo son la obtención de dos nuevos esquemas detectores de borde y dos métodos de restauración que han sido planteados, lo cuales se detallan a continuación.

- El primer detector es capaz de encontrar en imágenes ideales sintéticas la orientación, curvatura y localización sub-pixel de un contorno de forma exacta, así como el salto de intensidad producido a ambos lados del borde. Posteriormente, y a partir de dicho esquema, se ha desarrollado un algoritmo (ver función *Contornos2* en la página 109) para localizar los contornos sobre una imagen real, asumiendo que localmente dichos contornos se comportan como curvas de segundo grado. En base a las pruebas realizadas podemos afirmar que con este algoritmo se obtiene una gran precisión en el cálculo de los parámetros del borde.

- El segundo detector es capaz de encontrar también en imágenes ideales sintéticas, pero previamente convolucionadas con una máscara de suavizado de radio 1, los mismos parámetros del contorno de forma exacta. Posteriormente, y a partir de dicho esquema, se ha desarrollado un algoritmo (ver función *ContornosSuavizados* en la página 142) para localizar los contornos sobre una imagen real, cuyo funcionamiento es bastante bueno aunque la imagen tenga algo de ruido, e incluso aunque los contornos estén ligeramente desenfocados, como se ha visto en las pruebas realizadas.

- El primer método de restauración (ver funciones *RealzarImagen* en la página 161 y *ActualizarImagenes* en la página 197) consigue restaurar imágenes con gran cantidad de ruido, combinando un suavizado tradicional con la generación de subimágenes sintéticas a partir de la información obtenida por nuestro detector. Dicho método de restauración genera una imagen final de mayor calidad a la cual se le aplica nuestro detector para obtener los parámetros de los contornos presentes en la imagen original. En las pruebas hemos podido apreciar cómo la precisión en los valores de la detección conseguía ser bastante alta después de unas pocas iteraciones.
- El segundo método de restauración (ver función *ActualizarImagenes* en la página 207), más costoso que el anterior, es capaz de respetar contornos con alta curvatura, manteniéndolos lo más intactos posible durante las sucesivas iteraciones, así como evitar que dos contornos que estén muy cercanos entre sí vayan fusionándose durante el proceso. Los resultados realizados con imágenes angiográficas reales han mostrado cómo la detección de los contornos en zonas con este tipo de configuración ha sido bastante fiable.

11.1.2 Resultados en imágenes 3D

Para el caso de imágenes 3D, se ha repetido todo el estudio bidimensional extrapolándolo a tres dimensiones, con lo cual se han podido obtener también dos nuevos detectores de borde (superficies en este caso) y dos métodos de restauración, que se detallan a continuación:

- El primer detector es capaz de encontrar en imágenes ideales sintéticas el vector normal, los valores y direcciones de las curvatura principales, y la localización sub-voxel de un contorno de forma exacta, así como el salto de intensidad producido a ambos lados del borde. Posteriormente, y a partir de dicho esquema, se ha desarrollado un algoritmo (ver función *ContornosCurvos3D* en la página 266) para localizar los contornos sobre una imagen real, asumiendo que localmente dichos contornos se comportan como superficies de segundo grado. La precisión obtenida con este algoritmo es bastante alta, tal y como se ha podido ver en los ejemplos sintéticos realizados.
- El segundo detector es capaz de encontrar también en imágenes ideales sintéticas, pero previamente convolucionadas con una máscara de suavizado de radio 1, los mismos parámetros del contorno de forma exacta. Posteriormente, y a partir de dicho esquema, se ha desarrollado un algoritmo (ver función *ContornosSuavizados3D* en la página 301) para localizar los contornos sobre una imagen real, cuyo funcionamiento es bastante bueno aunque la imagen tenga algo de ruido, e incluso aunque los contornos estén ligeramente difuminados, como se ha podido ver en las pruebas realizadas.
- El primer método de restauración (ver función *RestaurarImagen3D* en la página 311) consigue restaurar imágenes con gran cantidad de ruido, combinando un suavizado tradicional con la generación de subimágenes sintéticas a partir de la información obtenida por nuestro detector. Dicho método de restauración genera una imagen final de mayor calidad a la cual se le aplica nuestro detector para obtener los parámetros de los contornos presentes en la imagen original. En las pruebas se ha podido constatar cómo la precisión se va mejorando en cada iteración.
- El segundo método de restauración (ver función *RestaurarImagen3D* en la página 338), más costoso que el anterior, es capaz de respetar superficies con altas curvaturas, manteniéndolas lo más intactas posible durante las sucesivas iteraciones, así como evitar la fusión de superficies muy cercanas entre sí. Los resultados realizados con imágenes reales han mostrado la gran precisión de los valores obtenidos.

11.2 Líneas futuras

El trabajo de investigación realizado, lejos de quedar finalizado, da pie a nuevos conceptos e ideas, y deja abiertas nuevas líneas de trabajo. Separando el caso de las imágenes 2D y 3D, las principales líneas abiertas

podrían ser las siguientes:

11.2.1 Líneas futuras para imágenes 2D

- El modelo de adquisición que hemos propuesto en este trabajo suponía el uso de una cámara de tipo pin-hole, donde cada punto del espacio tridimensional que resulta fotografiado se proyecta sobre un único punto del plano de proyección. Sin embargo, vimos en la sección 4.8 que es más realista suponer un **modelo de lente delgada**, en el cual algunos contornos pueden estar desenfocados en la imagen. El perfil de un borde ideal con este modelo de cámara se deduce en el anexo B. Por tanto, una ampliación del presente trabajo pudiera ser el rehacer todo el trabajo considerando este nuevo perfil.

- Nuestro trabajo se ha centrado en obtener los parámetros característicos del borde de forma individual para cada pixel. Sin embargo, existen muchas aplicaciones que precisan obtener la **línea continua** de todo el borde que represente la silueta de algún objeto presente en la imagen. Para ello, podría diseñarse un algoritmo que, a partir de la información obtenida en cada pixel y en sus vecinos, fuese agrupando éstos formando una línea continua de pixels. Aunque la secuencia de pixels formase una línea discreta, con la información de nuestro detector podría deducirse una expresión para una línea continua que representase todo el contorno con precisión real.

- En la misma línea, podría también desarrollarse un método de **segmentación**. La segmentación puede definirse como el proceso que divide una imagen en regiones u objetos cuyos píxels poseen atributos similares. Para lograr este objetivo podría partirse de nuestro esquema de restauración propuesto. Téngase en cuenta que en cada iteración de nuestro método ocurren dos procesos: por un lado, la zona de pixels perteneciente al borde, la cual se vendría a corresponder con las fronteras de las regiones de la segmentación, se va haciendo más nítida. Por otro lado, los pixels alejados de los bordes, los cuales se corresponderían con el interior de las regiones de la segmentación, se van haciendo cada vez más homogéneos. Atendiendo a este comportamiento, podría modificarse el método de forma que las iteraciones fueran convergiendo a segmentación final de la imagen.

- La **detección de esquinas** es otro proceso que suele ser necesario en muchas aplicaciones. Aunque directamente no hayamos tocado este tema en el presente trabajo, sí podría desarrollarse un detector de esquinas a partir de nuestro detector de bordes. Una esquina puede considerarse como una intersección entre dos líneas rectas. Por tanto lo primero que debería hacerse sería tratar de detectar rectas, lo cual podría hacerse buscando conjuntos de pixels borde vecinos con orientación y posición sub-pixel similar. Una vez obtenidas las rectas, a partir de sus expresiones hallaríamos las coordenadas exactas de su intersección.

- La **compresión de imágenes** es otro proceso bastante importante, ya que la cantidad de información a almacenar o transmitir es crucial en muchas aplicaciones. Así han surgido formatos como JPG o GIF, tan habituales de ver en las páginas de internet. Con las ideas planteadas en nuestra investigación podría plantearse un algoritmo de compresión. Para ello hay que tener en cuenta que, atendiendo al método de restauración que hemos propuesto, en cada iteración se genera una nueva imagen sintética a partir de dos informaciones diferentes: por un lado la imagen suavizada de la iteración anterior, y por otro lado, la información obtenida a partir del detector de bordes. Sin embargo, puede verse que la información relevante para generar una nueva imagen es esta última, ya que en ella se incluye también una estimación del valor de la intensidad a cada lado del borde en cada pixel. Por tanto, podría plantearse un algoritmo que, partiendo de la información de los contornos en cada pixel borde, generase pequeñas subimágenes sintéticas alrededor de cada pixel borde (al igual que se hace en nuestra restauración propuesta), y luego fuese extendiéndose a los pixels vecinos hasta cubrir toda la imagen por completo. Si esto se logra, significaría que para almacenar la imagen podríamos solamente almacenar la información obtenida por

nuestro detector en aquellos pixels considerados borde, la cual representará un volumen de información mucho menor que la totalidad de los valores de los pixels. Se necesitaría finalmente realizar un estudio del grado de compresión obtenido, para poder compararlo con las técnicas tradicionales.

- Otro de los problemas críticos en visión artificial es la necesidad de **incrementar la resolución** espacial de las imágenes, especialmente en aquellas zonas de mayor relevancia para su análisis (zonas con bordes). Para ello se han propuesto sistemas que se basan en la modelización de los bordes a partir de la información en niveles de gris de la imagen[1]. Como nuestro método detector es capaz de obtener información precisa sobre el contorno en cada pixel, podría generarse una nueva imagen sintética, de la misma forma que lo hace nuestro método de restauración, pero a una resolución mayor, es decir, generando para cada pixel una imagen de nxn subpixels.

- Centrándonos en las **imágenes angiográficas**, faltaría hacer un estudio más en profundidad de la comparación de los resultados con otros métodos detectores, como los comentados en la sección 0.3.3, y utilizar un conjunto de imágenes más extenso.

11.2.2 Líneas futuras para imágenes 3D

- El método de restauración propuesto para altas curvaturas es bastante costoso computacionalmente, por lo que para imágenes muy grandes puede ser excesivamente lento. Hay que tener en cuenta que en cada iteración se debe estimar en cada voxel borde un paraboloide rotado, y para ello se plantea una función iterativa que va generando varias subimágenes sintéticas de paraboloides hasta lograr la convergencia (ver función *EstimarParaboloideRotado* en la página 337). A su vez, para cada subimagen se plantean varias funciones de integración recursivas (ver anexo C) que van estimando la intensidad de cada voxel. Sería conveniente realizar una **optimización** de todos estos procesos para disminuir los tiempos de ejecución.

- En la sección 10.4.2 se analizó que una **aproximación por toroides** en lugar de por paraboloides sería más exacta. Queda abierto un estudio más en profundidad de la viabilidad de poder desarrollar un esquema que use toroides y que no sea excesivamente costoso de implementar.

- Nuestro algorimo detector obtiene información en cada voxel borde de los parámetros de la superficie del contorno que pasa por dicho voxel. Sin embargo, no genera por sí solo una **superficie continua** que represente todo el contorno, tal y como hacen otros métodos (marching cubes, isosuperficies, etc). A partir de nuestro esquema detector podría plantearse un algoritmo que produjera una superficie cerrada. De esta forma la visualización de los resultados sería mucho más realista.

- De igual forma, podría deducirse un método de **segmentación** que divida en regiones homogéneas toda la imagen. Al igual que comentamos en el apartado anterior, podríamos lograrlo partiendo de nuestro método de restauración propuesto.

- En cuanto a la **compresión de imágenes**, puede rehacerse también el mismo comentario que se hizo para imágenes 2D, y tratar de plantear un algoritmo que, partiendo solamente de la información obtenida por el detector, genere una imagen sintética lo más parecida a la original.

- Centrándonos en las **imágenes angiográficas**, muchos de los métodos expuestos en la sección 0.3.3 tratan de buscar la línea central de los vasos o *centerline*. Podría plantearse un algoritmo que buscase esta línea a partir de la información obtenida por nuestro detector. Por ejemplo, el vector normal en cada voxel borde nos indica la dirección aproximada en la que se encontrará la línea central. La magnitud de la curvatura máxima nos da una idea de la distancia a la que se encuentra. Por otro lado, si nos movemos por los

[1] http://lorca.umh.es/isa/es/proyectos/subpixel

voxels vecinos atendiendo a la variación de la orientación de la superficie, parece que podría obtenerse una estimación bastante precisa de la ubicación del centerline.

Anexo A

Probabilidad de error del método cuadrático en función del radio de curvatura

Consideremos la figura A.1, donde una circunferencia de radio R, con centro en el punto $C = (x_c, y_c)$, atraviesa el pixel central de la ventana de tal forma que corta a la vertical central de dicho pixel en el punto $P = (0, d)$. Supongamos que d puede tomar cualquier valor del intervalo $[-h/2, h/2]$ con idéntica probabilidad. De igual manera, supongamos que el ángulo α que posee la recta tangente a la circunferencia en P puede tomar cualquier valor del intervalo $[0, \pi/4]$ también con idéntica probabilidad. Estamos interesados en calcular la probabilidad de que la circunferencia no toque el margen izquierdo de una ventana $5x3$ centrada en dicho pixel, es decir, que no toque la línea vertical $x = -3h/2$ (o al menos, no por encima de la altura $y = -5h/2$).

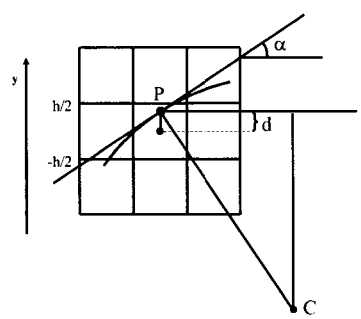


Figura A.1: La circunferencia con centro C corta la vertical central del pixel central en el punto $P = (0, d)$.

Sabemos que para un valor de d y de α concreto, el centro de la circunferencia estará en el punto

$$\begin{aligned} x_c &= R \sin \alpha \\ y_c &= d - R \cos \alpha \end{aligned}$$

y por lo tanto la expresión de la curva que cruza el pixel central vendrá dada por la expresión:

$$f(x) = y_c + \sqrt{R^2 - (x - x_c)^2}$$

Para que dicha circunferencia no alcance el margen izquierdo de la ventana $5x3$ debe cumplirse que $f(-3h/2) > -5h/2$, con lo cual se deduce que

$$d > R\cos\alpha - \frac{5}{2}h - \sqrt{R^2 - \left(\frac{3}{2}h + R\sin\alpha\right)^2} \tag{A.1}$$

Llamemos a esta expresión $y(R,\alpha)$. En la figura A.2 podemos ver varias curvas de $y(R,\alpha)$ para varios valores de R, dibujadas sobre el plano de las variables α y d. El rectángulo pintado en verde representa todos los casos posibles que queremos tratar. Así por ejemplo, para $R = 20$ vemos que $y(20,\alpha)$ no alcanza el rectángulo, y por tanto, para todos los casos posibles de d y α, se cumplirá siempre la inecuación A.1. Por el contrario, para $R = 6$, existirían algunos casos en donde la expresión A.1 no se cumpliría, y por tanto la probabilidad de que esto ocurra vendría dada por la relación entre el área interior al rectángulo que cae bajo la curva $y(6,\alpha)$ y el área total del rectángulo, $\pi h/4$.

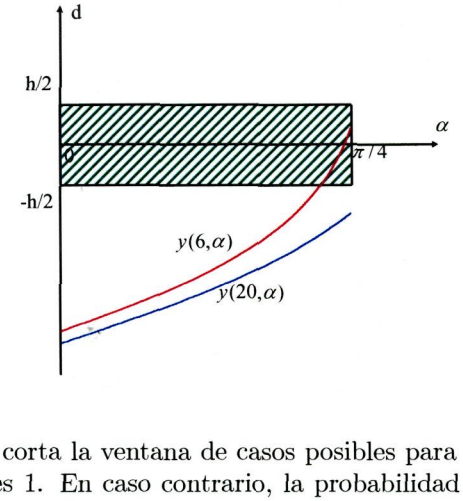


Figura A.2: Si la curva $y(R,\alpha)$ no corta la ventana de casos posibles para ningún valor de α entre 0 y $\pi/4$, la probabilidad de no cometer error es 1. En caso contrario, la probabilidad está en función del área interior al rectángulo sobre la curva.

Para obtener una expresión completa de la función de probabilidad, $P(R)$, habrá entonces que estudiar cada uno de los casos posibles.

A.1 Casos posibles

A.1.1 Caso R>>

Para valores grandes del radio está claro que la curva $y(R,\alpha)$ jamás tocará el rectángulo de casos posibles, y por tanto, ninguna circunferencia que corte la vertical del pixel central con un ángulo entre 0 y $\pi/4$ podrá alcanzar la línea $x = -3h/2$. Esto significa que la probabilidad de no cometer error en este caso es 1.

Para valores más pequeños de R, la curva $y(R,\alpha)$ va acercándose cada vez más al rectángulo, llegándolo a tocar para el valor de R que cumpla que $y(R,\pi/4) = -h/2$. Desarrollando esta expresión llegamos a la conclusión de que esto ocurrirá cuando

$$R = \frac{25}{4}\sqrt{2}h \simeq 8.84h$$

que coincide con el valor deducido en la sección 3.3.3 (ecuación 3.9).

A.1.2 Caso $R < 8.84h$

Para valores ligeramente inferiores a $8.84h$, la situación se muestra en la figura A.3 donde la curva $y(R, \alpha)$ corta al rectángulo verde en sus lados inferior y derecho.

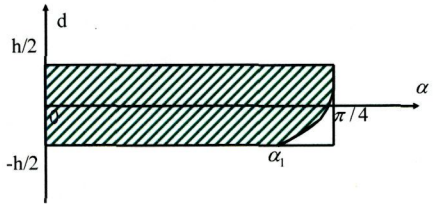


Figura A.3: La curva $y(R, \alpha)$ corta al rectángulo en sus lados inferior y derecho.

La probabilidad por tanto será el área relativa del rectángulo que cae sobre la curva. Es decir

$$P_1(R) = 1 - \frac{4}{\pi h} \int_{\alpha_1}^{\pi/4} \left(y(R, \alpha) + \frac{h}{2} \right) d\alpha$$

donde α_1 indica el corte de la curva con la base del rectángulo, cuya expresión viene dada por:

$$\alpha_1 = \arcsin \left(\frac{1}{400R^2} \left(-300Rh + 80\sqrt{16R^4 - 25R^2h^2} \right) \right) \tag{A.2}$$

Como dicha integral es irresoluble analíticamente, podemos aproximarla por el área de un triángulo, con lo cual la probabilidad quedará con la siguiente expresión

$$\begin{aligned} P_1(R) &\approx 1 - \frac{2}{\pi h} \left(\frac{\pi}{4} - \alpha_1 \right) \left(y(R, \frac{\pi}{4}) + \frac{h}{2} \right) = \\ &= \frac{3}{4} + \frac{1}{\pi}\alpha_1 + \frac{1}{2\pi h} (4\alpha_1 - \pi)\, y(R, \frac{\pi}{4}) \end{aligned}$$

donde α_1 también depende de R.

Este caso ocurrirá hasta que $y(R, \alpha)$ toque el lado superior del rectángulo, lo cual pasará cuando se cumpla que $y(R, \pi/4) = h/2$, es decir,

$$R = \frac{15}{4}\sqrt{2}h \simeq 5.30h$$

A.1.3 Caso $R < 5.30h$

En este caso la curva $y(R, \alpha)$ corta el rectángulo en sus lados inferior y superior, como se muestra en la figura A.4.

La probabilidad para este caso vendrá dada por la función

$$P_2(R) = \frac{4}{\pi h} \left(\alpha_1 h + \int_{\alpha_1}^{\alpha_2} \left(y(R, \alpha) + \frac{h}{2} \right) d\alpha \right)$$

siendo α_1 igual a la expresión A.2, y α_2 igual a la siguiente expresión:

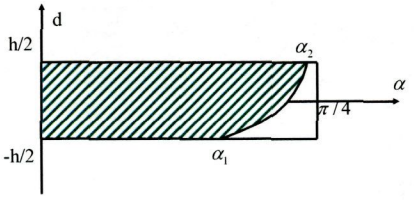


Figura A.4: La curva $y(R, \alpha)$ corta el rectángulo en sus lados inferior y superior.

$$\alpha_2 = \arcsin\left(\frac{1}{80R^2}\left(-60Rh + 8\sqrt{5}\sqrt{16R^4 - 45R^2h^2}\right)\right) \tag{A.3}$$

Aproximando de nuevo la integral de forma lineal obtenemos la expresión aproximada para P_2 :

$$P_2(R) \approx \frac{4}{\pi h}\left(\alpha_1 h + \frac{1}{2}h\left(\alpha_2 - \alpha_1\right)\right) = \frac{2}{\pi}\left(\alpha_1 + \alpha_2\right)$$

Téngase en cuenta que la función $y(R, \alpha)$ no está definida para todos los valores de α. En realidad, su dominio viene dado por la inecuación

$$R^2 - \left(\frac{3}{2}h + R\sin\alpha\right)^2 > 0$$

lo cual ocurrirá para valores de α que cumplan que

$$\alpha < \arcsin\left(1 - \frac{3h}{2R}\right)$$

Esto significa que este caso acabará cuando el límite del dominio coincida con α_2, y esto ocurrirá cuando se cumpla:

$$R = \frac{15}{4}h \simeq 3.75h$$

A.1.4 Caso $R < 3.75h$

En este caso la curva corta el rectángulo en su lado inferior, y luego deja de estar definida, como se muestra en la figura A.5.

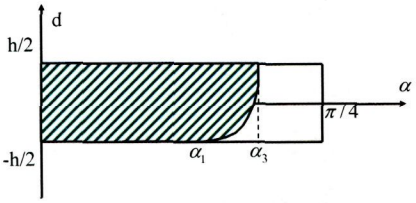


Figura A.5: La curva $y(R, \alpha)$ corta el rectángulo en su lado inferior, y luego deja de estar definida.

La probabilidad para este caso vendrá dada por la función

$$P_3(R) = 1 - \frac{4}{\pi h}\left(\left(\frac{\pi}{4} - \alpha_3\right)h + \int_{\alpha_1}^{\alpha_3}\left(y(\alpha) + \frac{h}{2}\right)d\alpha\right)$$

siendo α_1 igual a la expresión A.2, y α_3 igual a la siguiente expresión:

$$\alpha_3 = \arcsin\left(1 - \frac{3h}{2R}\right) \tag{A.4}$$

Aproximando de nuevo la integral de forma lineal obtenemos la expresión aproximada para P_3 :

$$\begin{aligned} P_3(R) &\approx 1 - \frac{4}{\pi h}\left(\left(\frac{\pi}{4} - \alpha_3\right)h + \frac{1}{2}(\alpha_3 - \alpha_1)\left(y(R,\alpha_3) + \frac{h}{2}\right)\right) = \\ &= \frac{1}{\pi}(\alpha_1 + 3\alpha_3) + \frac{2}{\pi h}(\alpha_1 - \alpha_3)\, y(R,\alpha_3) \end{aligned}$$

Este caso ocurrirá hasta que $y(R,\alpha)$ deje de estar definido en el interior del rectángulo, y esto sucederá cuando el límite del dominio coincida con α_2, es decir, cuando

$$R = \frac{25}{12}h \simeq 2.08h$$

A.1.5 Caso $R < 2.08h$

En este caso la curva no toca el rectángulo, con lo cual la probabilidad viene dada por el rectángulo limitado por los valores 0 y α_3, valor este último donde deja de estar definida la curva (figura A.6).

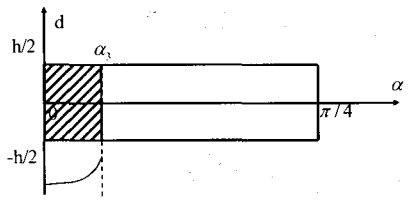


Figura A.6: La curva $y(R,\alpha)$ no toca el rectángulo.

La probabilidad para este caso vendrá dada por la función

$$P_4(R) = \frac{4}{\pi}\alpha_3$$

siendo α_3 igual a la expresión A.4.
Este caso ocurrirá hasta que $y(R,\alpha)$ deje de estar definida, y esto ocurrirá cuando $R = 3h/2$.

A.1.6 Caso $R < 1.5h$

En este caso la probabilidad de no cometer error es cero, ya que es imposible que un contorno que cruce el pixel central con pendiente entre 0 y 1 alcance la línea vertical $x = -3h/2$.

A.2 Probabilidad final

Combinando todos los casos llegamos a la expresión final aproximada para la probabilidad de no cometer un error con nuestro método detector en función del radio de curvatura del contorno:

$$P(R) = \begin{cases} 0 & 0 < R < \frac{3}{2}h \\ \frac{4}{\pi}\alpha_3 & \frac{3}{2}h < R < \frac{25}{12}h \\ \frac{1}{\pi}(\alpha_1 + 3\alpha_3) + \frac{2}{\pi h}(\alpha_1 - \alpha_3)\, y(R, \alpha_3) & \frac{25}{12}h < R < \frac{15}{4}h \\ \frac{2}{\pi}(\alpha_1 + \alpha_2) & \frac{15}{4}h < R < \frac{15}{4}\sqrt{2}h \\ \frac{3}{4} + \frac{1}{\pi}\alpha_1 + \frac{1}{2\pi h}(4\alpha_1 - \pi)\, y(R, \frac{\pi}{4}) & \frac{15}{4}\sqrt{2}h < R < \frac{25}{4}\sqrt{2}h \\ 1 & \frac{25}{4}\sqrt{2}h < R \end{cases}$$

donde α_1, α_2 y α_3 vienen dados por las expresiones A.2, A.3 y A.4 respectivamente. La gráfica de la función se muestra en la figura A.7.

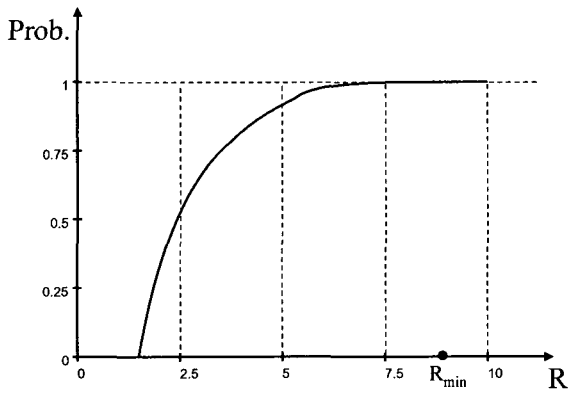


Figura A.7: Probabilidad de que una circunferencia de radio R no produzca error con nuestro método. El radio está medido en unidades de h.

Anexo B

Perfil de un borde ideal con el modelo de lente delgada

Tomemos un contorno vertical, como el de la figura B.1, que separa dos niveles de intensidad, A y B a ambos lados. Con el modelo pin-hole, cada punto del espacio es proyectado en un único punto del plano imagen. Se produce por tanto una discontinuidad en la señal $f(x, y)$ recibida en el sensor. Posteriormente la discretización produce una señal $F(x_i, y_j)$ con una única columna con intensidad intermedia entre A y B.

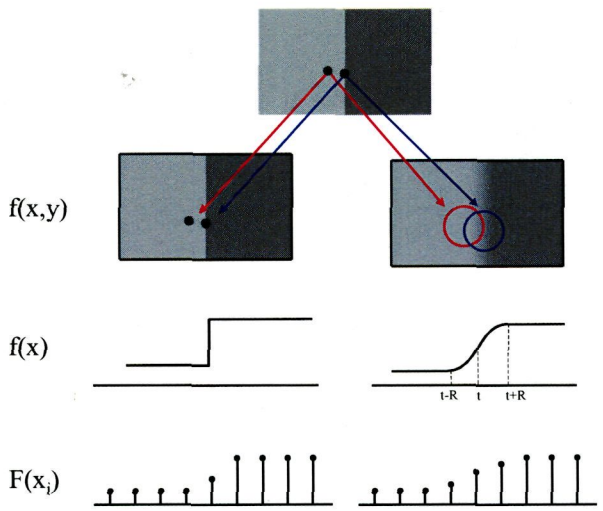


Figura B.1: Formación de la imagen con el modelo de pin-hole (columna izquierda) y con el modelo de lente delgada (columna derecha). De arriba a abajo: Contorno vertical en el espacio. Intensidad que llega a la cámara, $f(x, y)$. La misma vista en una sola fila de la imagen. Imagen discretizada.

Por el contrario, con el modelo de lente delgada, cada punto del espacio se proyecta en una zona circular sobre el plano imagen, con lo cual cada punto (x, y) de la imagen recibe intensidad de una zona equivalente en el espacio. Esto provoca una difuminación en la señal, produciendo una señal discretizada con varias columnas de intensidad intermedia entre A y B. Tratemos de obtener pues una expresión para $f(x)$, tomando sólamente un caso unidimensional.

Tomemos el caso representado en la figura B.2, donde en la parte superior se muestra un contorno vertical ideal localizado en la posición $x = t$. En la parte inferior se muestra la intensidad adquirida en la cámara. Estamos interesados en obtener la intensidad que llega a un punto concreto $x = x_0$. Sabemos que a ese punto llegará intensidad procedente de un área circular sobre la escena que se quiere fotografiar. Por tanto, la intensidad buscada no es más que un promedio entre las intensidades A y B ponderadas por el área relativa interior al círculo de cada una de ellas, y su expresión vendrá dada por:

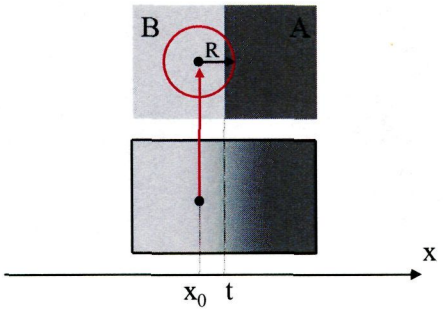


Figura B.2: De arriba a abajo: Contorno vertical en el espacio. Intensidad que llega a la cámara, $f(x)$.

$$f(x_0) = \begin{cases} B & x_0 \leq t - R \\ B + (A - B)\, I & t - R < x_0 < t + R \\ A & x_0 \geq t + R \end{cases}$$

donde I indica el área relativa de círculo que cae en la parte correspondiente a la intensidad A, cuya expresión es

$$I = \frac{2}{\pi R^2} \int_t^{x_0+R} \sqrt{R^2 - (x - x_0)^2}\, dx$$

Resolviendo esta integral obtenemos que

$$I = \frac{1}{2} - \frac{1}{\pi} \arcsin\left(\frac{t - x_0}{R}\right) - \frac{1}{\pi} \frac{t - x_0}{R} \sqrt{R^2 - (t - x_0)^2}$$

con lo cual, generalizando el valor x_0 para cualquier valor x, obtenemos la expresión final para el perfil que estábamos buscando:

$$f(x) = \begin{cases} B & x < t - R \\ \frac{A+B}{2} - \frac{A-B}{\pi} \left(\arcsin u + u\sqrt{1 - u^2}\right) & t - R < x < t + R \\ A & x > t + R \end{cases}$$

donde

$$u = \frac{t - x}{R}$$

La gráfica de dicha función puede verse en la figura B.3.

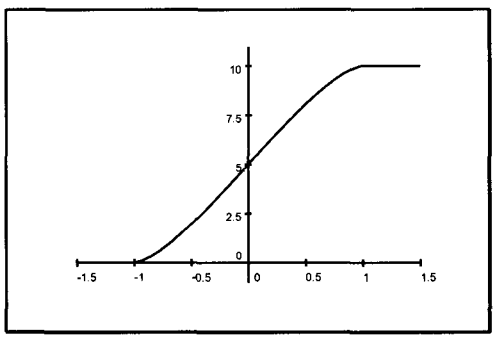


Figura B.3: Perfil de un borde con el modelo de lente delgada para el caso $A = 10$, $B = 0$ y $R = 1$

Anexo C

Cálculo de volúmenes

C.1 Volumen bajo un plano interior a un prisma

Consideremos el prisma de la figura C.1, dado por las esquinas (x_l, y_d, z_b) y (x_r, y_u, z_f). Dicho prisma puede representar tanto a un voxel completo como a un sub-voxel. De ahí que lo planteemos como un prisma general.

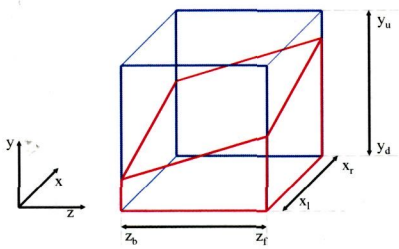


Figura C.1:

Sea el plano $y = a + bx + cz$, donde supondremos el caso en que $1 \geq b \geq c \geq 0$. Es decir, teniendo en cuenta que la normal al plano viene dada por la dirección $[b, -1, c]$, lo que estamos haciendo es quedarnos con el caso en el que la componente y del vector normal es mayor en valor absoluto, y por tanto, está más cercana a la dirección del eje y que a los otros dos ejes.

El objetivo es calcular el volumen bajo el plano interior al prisma. Pero para poder plantear la integral doble que nos dé el volumen es preciso conocer la intersección del plano con el suelo y el techo del prisma. Existen 8 configuraciones distintas en las que un plano como el que nos ocupa intersecta el prisma, que son las que se muestran en la figura C.2.

Para diferenciar cada caso basta ver cuál es la altura del plano sobre cada una de las cuatro esquinas del prisma. Calculando cuántas de ellas se encuentran bajo el prisma, cuántas en el interior, y cuántas sobre él, se deduce en cuál de las 8 configuraciones nos encontramos.

Antes de plantear la expresión del volumen en cada una, definamos las tres siguientes funciones básicas de integración, acorde con la figura C.3.

C.1.1 Integración punto a punto

El caso PP de la figura C.3 muestra un recinto de integración rectangular, donde se quiere evaluar la integral

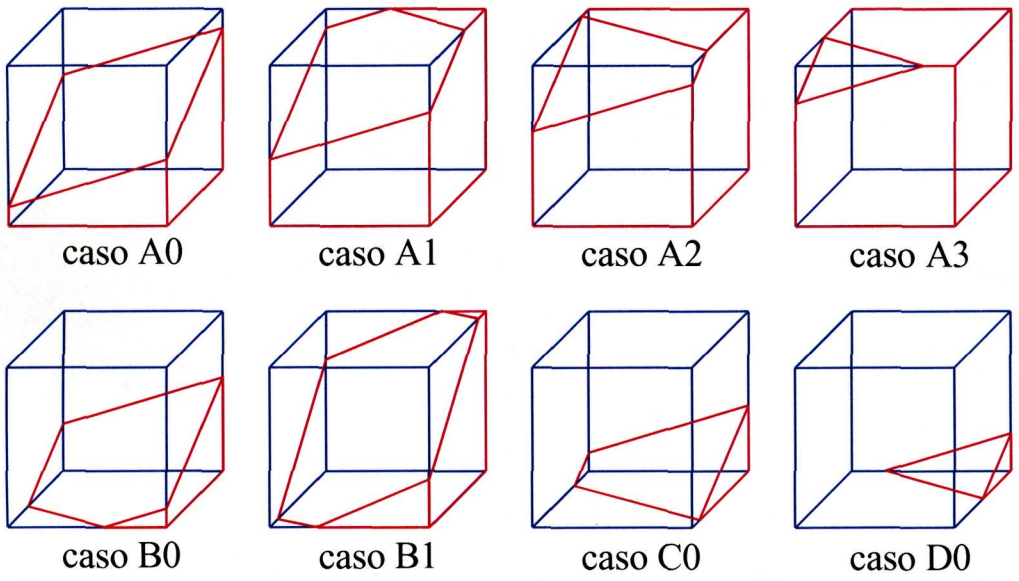


Figura C.2: Existen 8 casos diferentes en los que el plano $y = a + bx + cz$, tal que $1 \geq b \geq c \geq 0$ corta un prisma. La letra del caso indica cuántas de las 4 esquinas se encuentran bajo el prisma (A:0, B:1, C:2, D:3). El número indica cuántas de las 4 esquinas se encuentran por encima del prisma

$$I = \int_{z_0}^{z_1} \int_{x_0}^{x_1} (a + bx + cz) dx dz$$

Esta integral viene dada por la expresión

$$I = \frac{1}{2} (z_1 - z_0) (x_1 - x_0) (2a + bx_0 + bx_1 + cz_0 + cz_1)$$

El pseudo código sería el siguiente:

Funcion Int1PuntoPunto (z0, z1, x0, x1, a, b, c)

```
Retornar (0.5 * (z1-z0) * (x1-x0) * (2*a+b*x0+b*x1+c*z0+c*z1))
```

C.1.2 Integración punto a recta

El caso PR de la figura C.3 muestra un recinto donde se quiere evaluar la integral

$$I = \int_{z_0}^{z_1} \int_{x_0}^{mz+d} (a + bx + cz) dx dz$$

siendo $mz + d$ la recta que pasa por los puntos (x_2, z_2) y (x_3, z_3). La integral interior puede expresarse como

$$\begin{aligned} \int_{x_0}^{mz+d} (a + bx + cz)\, dx &= \left(a\,(d - x_0) + \frac{1}{2} b\left(d^2 - x_0^2\right)\right) + (cd + am + bdm - cx_0)\, z + \left(cm + \frac{1}{2} bm^2\right) z^2 \\ &= A + Bz + Cz^2 \end{aligned}$$

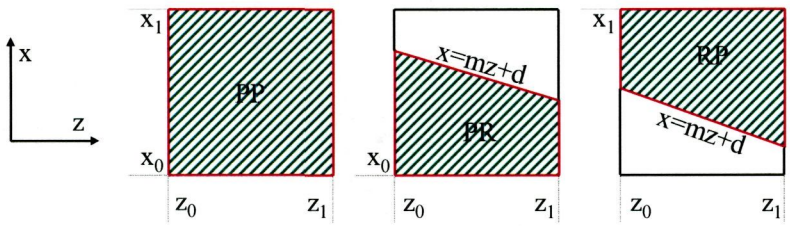


Figura C.3: Los tres tipos de recinto de integración que se necesitan: PP (punto a punto), PR (de punto a recta) y RP (de recta a punto)

Por tanto la integral exterior se convierte en

$$I = \int_{z_0}^{z_1} \left(A + Bz + Cz^2\right) dz = A\left(z_1 - z_0\right) + \frac{1}{2}B\left(z_1^2 - z_0^2\right) + \frac{1}{3}C\left(z_1^3 - z_0^3\right) \tag{C.1}$$

El pseudo código sería el siguiente:

Funcion Int1PuntoRecta (z0, z1, x0, x2, z2, x3, z3, a, b, c)

```
// calculamos las potencias.....................
x02=x0*x0;
z02=z0*z0; z03=z02*z0
z12=z1*z1; z13=z12*z1
// calculamos la recta..........................
m = (x3-x2) / (z3-z2)
d = x2 - m*z2
m2 = m * m
d2 = d * d
// calculamos la integral.....................
A = a*(d-x0) + 0.5*b*(d2-x02)
B = c*d + a*m + b*d*m - c*x0
C = c*m + 0.5*b*m2
Retornar (A*(z1-z0) + 0.5*B*(z12-z02) + 1/3*C*(z13-z03))
```

C.1.3 Integración recta a punto

El caso RP de la figura C.3 muestra un recinto donde se quiere evaluar la integral

$$I = \int_{z_0}^{z_1} \int_{mz+d}^{x_1} (a + bx + cz) dx dz$$

siendo $mz + d$ la recta que pasa por los puntos (x_2, z_2) y (x_3, z_3). La integral interior puede expresarse como

$$\begin{aligned} \int_{mz+d}^{x_1} (a + bx + cz)\, dx &= \left(a\left(x_1 - d\right) + \frac{1}{2}b\left(x_1^2 - d^2\right)\right) + \left(c\left(x_1 - d\right) - m\left(a + bd\right)\right) z + \left(-cm - \frac{1}{2}bm^2\right) z^2 \\ &= A + Bz + Cz^2 \end{aligned}$$

La integral exterior coincide con la expresión C.1. El pseudo código sería el siguiente:

Funcion Int1RectaPunto (z0, z1, x2, z2, x3, z3, x1, a, b, c)

```
// calculamos las potencias..................
x12=x1*x1;
z02=z0*z0; z03=z02*z0
z12=z1*z1; z13=z12*z1
// calculamos la recta.......................
m = (x3-x2) / (z3-z2)
d = x2 - m*z2
m2 = m * m
d2 = d * d
// calculamos la integral....................
A = a*(x1-d) + 0.5*b*(x12-d2)
B = c*(x1-d) - m*(a+b*d)
C = -c*m - 0.5*b*m2
Retornar (A*(z1-z0) + 0.5*B*(z12-z02) + 1/3*C*(z13-z03))
```

C.1.4 Cálculo del volumen

En la figura C.4 se muestran los recintos de integración para cada uno de los 8 casos. Atendiendo a cada uno de ellos, habrá que llamar a las funciones anteriores específicas para cada caso.

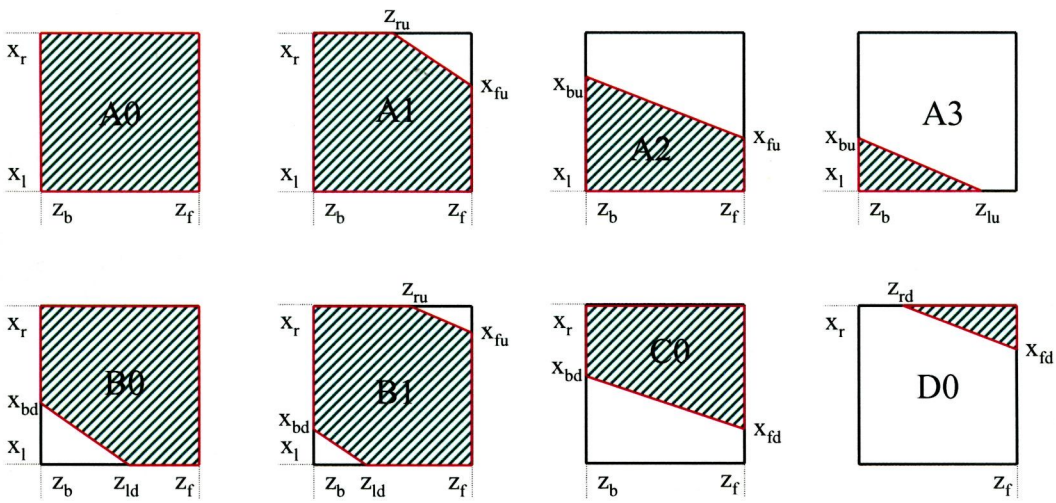


Figura C.4: Recintos de integración para los 8 casos

Simetrías Por otro lado, recordemos que estamos tratando sólo con planos de la forma $y = a + bx + cz$ donde se cumple que $1 \geq b \geq c \geq 0$. Para tratar la situación más general dónde sólamente se cumpla que $1 \geq |b|, |c| \geq 0$ simplemente habrá que modificar el valor de los coeficientes del plano para convertirlos en este caso inicial.

Por ejemplo, supongamos que $b < 0$. Consideremos el punto (x_0, y_0, z_0) perteneciente al plano tal que (x_0, z_0) indica la vertical central del prisma, e y_0 indica el corte del plano con dicha vertical. Ya hemos dicho que la

normal al plano es el vector $[b, -1, c]$. Por tanto, el volumen bajo el plano es equivalente al volumen de un segundo plano que pase por el mismo punto (x_0, y_0, z_0), y cuya normal sea $[-b, -1, c]$.

Para obtener la expresión de dicho plano, hacemos un cambio de variable $u = x - x_0$, $v = y - y_0$, $w = z - z_0$, con lo cual el plano queda

$$\begin{aligned} v + y_0 &= a + b\,(u + x_0) + c\,(w + z_0) \\ v &= (a - y_0 + bx_0 + cz_0) + bu + cw = A + bu + cw \end{aligned}$$

A continuación invertimos el signo de u, y finalmente deshacemos el cambio:

$$\begin{aligned} v &= A - bu + cw \\ (y - y_0) &= A - b(x - x_0) + c(z - z_0) \\ y &= (A + y_0 + bx_0 - cz_0) - bx + cz = (a + 2bx_0) - bx + cz \end{aligned}$$

Por tanto, el nuevo plano es $y = a' + b'x + c'z$ donde

$$\begin{aligned} a' &= a + 2bx_0 \\ b' &= -b \\ c' &= c \end{aligned}$$

El caso $c < 0$ es similar. Finalmente, el caso $c > b > 0$ se corrige simplemente intercambiando los valores de los ejes x y z.

A continuación se muestra el pseudo-código final de la función que evalúa el volumen bajo un plano interior a un prisma.

Funcion VolumenVoxelPlano (xl, xr, yd, yu, zb, zf, a, b, c)

```
// nos centraremos en el caso b > c > 0 ..........................
Si b<0 Entonces b=-b; a-=b*(xl+xr)
Si c<0 Entonces c=-c; a-=c*(zb+zf)
Si b<c Entonces Intercambiar (b,xl,xr) <---> (c,zb,zf)
// deducir en que caso estamos ..................................
Calcular la altura del plano sobre las 4 esquinas (ylb, ylf, yrb, yrf)
Si ylb>=yu Retornar (1)
Si yrf<=yd Retornar (0)
Deducir en cual de los 8 casos estamos
// calcular el volumen ..........................................
Segun (caso)
   caso A0: vol = Int1PuntoPunto (zb, zf, xl, xr, a-yd, b, c)
   caso A1: zru = (yu-a-b*xr) / c
            xfu = (yu-a-c*zf) / b
            vol = Int1PuntoPunto (zb, zru, xl, xr, a-yd, b, c)
            vol += Int1PuntoRecta (zru, zf, xl, xr, zru, xfu, zf, a-yd, b, c)
            vol += 0.5 * (xr-xfu) * (zf-zru) * (yu-yd)
   caso A2: xbu = (yu-a-c*zb) / b
            xfu = (yu-a-c*zf) / b
            vol = Int1PuntoRecta (zb, zf, xl, xbu, zb, xfu, zf, a-yd, b, c)
            vol += (xr-0.5*(xbu+xfu)) * (zf-zb) * (yu-yd)
```

```
    caso A3: xbu = (yu-a-c*zb) / b
             zlu = (yu-a-b*xl) / c
             vol = Int1PuntoRecta (zb, zlu, xl, xbu, zb, xl, zlu, a-yd, b, c)
             vol += (xr-xl)*(yu-yd)*(zf-zb) - 0.5*(xbu-xl)*(zlu-zb)*(yu-yd)
    caso B0: xbd = (yd-a-c*zb) / b
             zld = (yd-a-b*xl) / c
             vol = Int1RectaPunto (zb, zld, xbd, zb, xl, zld, xr, a-yd, b, c)
             vol += Int1PuntoPunto (zld, zf, xl, xr, a-yd, b, c)
    caso B1: xbd = (yd-a-c*zb) / b
             zld = (yd-a-b*xl) / c
             zru = (yu-a-b*xr) / c
             xfu = (yu-a-c*zf) / b
             vol = Int1RectaPunto ( zb, zld, xbd, zb, xl, zld, xr, a-yd, b, c)
             vol += Int1PuntoPunto (zld, zru, xl, xr, a-yd, b, c)
             vol += Int1PuntoRecta (zru, zf, xl, xr, zru, xfu, zf, a-yd, b, c)
             vol += 0.5 * (xr-xfu) * (zf-zru) * (yu-yd)
    caso C0: xbd = (yd-a-c*zb) / b
             xfd = (yd-a-c*zf) / b
             vol = Int1RectaPunto (zb, zf, xbd, zb, xfd, zf, xr, a-yd, b, c)
    caso D0: zrd = (yd-a-b*xr) / c
             xfd = (yd-a-c*zf) / b
             vol = Int1RectaPunto (zrd, zf, xr, zrd, xfd, zf, xr, a-yd, b, c)
// retornamos el volumen relativo.....................................
v = (xr-xl) * (yu-yd) * (zf-zb)
Retornar (vol/v)
```

C.2 Volumen bajo un paraboloide interior a un prisma

Calcular el volumen interior al prisma que se encuentra bajo un paraboloide $y = a+bx+cz+dx^2+fxz+gz^2$ es una tarea bastante más compleja que calcular el volumen bajo un plano. Cuando la situación es la de la figura C.5a, donde el paraboloide no toca ni el suelo ni el techo del prisma, no hay ningún problema. Simplemente hay que deducir la integral doble siguiente:

$$I = \int_{z_b}^{z_f} \int_{x_l}^{x_r} (a + bx + cz + dx^2 + fxz + gz - y_d) dx dz$$

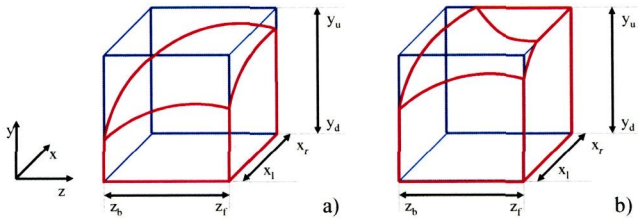


Figura C.5:

Sin embargo, cuando la superficie toca el suelo o el techo del prisma (ver figura C.5b), hay que tener en cuenta que la curva intersección del paraboloide con uno de esos planos produce una cónica rotada. Por ejemplo, el corte de la superficie con el techo produce la curva

$$y_u = a + bx + cz + dx^2 + fxz + gz^2$$

lo que da lugar a un recinto de integración bastante complejo de resolver.

Por otro lado, aunque hiciéramos una aproximación más simple suponiendo que los cortes con las caras del prisma fueran siempre rectas en lugar de cónicas, el número de casos posibles en los que un paraboloide puede cortar a un prisma son bastante más que para el caso cuando la superficie es un plano. Hay que tener en cuenta que cuando usábamos un plano, dado por la ecuación $y = a + bx + cz$, unido a la suposición de que $1 \geq b \geq c \geq 0$, reducíamos el número de casos posibles a 8 (figura C.2). Sin embargo, cuando usamos paraboloides, la situación en el interior del voxel tiene muchas más posibilidades, ya que, aunque nos centremos de nuevo en el caso $1 \geq b \geq c \geq 0$, por el hecho de tener curvaturas, los cortes con las caras del prisma pueden producirse en cualquier lugar. Por ejemplo, solamente el caso A1 de la figura C.2 tendría 4 subcasos, como los mostrados en la figura C.6.

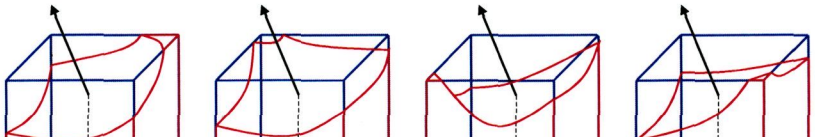


Figura C.6: Casos en los que un paraboloide $y = a + bx + cz + dx^2 + fxz + gz^2$ tal que $1 \geq b \geq c \geq 0$ tiene una sola esquina por encima del prisma

C.2.1 Aproximación por planos

Otra alternativa es utilizar la función **VolumenVoxelPlano()** que definimos en la sección anterior, y plantear un esquema recursivo que vaya subdividiendo el prisma en 8 subprismas mientras la curvatura del paraboloide esté por encima de un cierto umbral. Cuando la curvatura sea inferior, significará que la aproximación por un plano dará una estimación del volumen bastante preciso. Si no es así, seguiremos subdividiendo hasta alcanzar un prisma de tamaño mínimo. Este sistema no dará la solución exacta, pero al menos podemos acotar el error cometido según el valor umbral que prefijemos.

Antes de plantear este algoritmo, vamos a demostrar que cuando el paraboloide no toca ni el suelo ni el techo del prisma (caso A0), existe un plano equivalente donde el volumen del plano y el paraboloide son exactamente iguales. Sea por tanto el paraboloide dado por la ecuación $y = a + bx + cz + dx^2 + fxz + gz^2$, y el objetivo es calcular el volumen bajo dicho paraboloide sobre un recinto rectangular de dimensión $h_x \times h_z$ centrado en el punto (x_0, z_0). Para ello queremos encontrar el plano $y = a' + b'x + c'z$, con la misma normal sobre el punto (x_0, z_0) que el paraboloide, cuyo volumen sea el mismo (ver figura C.7).

Para que ambas normales coincidan sobre dicho punto es fácil demostrar que los coeficientes b' y c' han de tener la siguiente expresión:

$$\begin{aligned} b' &= b + 2dx_0 + fz_0 \\ c' &= c + fx_0 + 2gz_0 \end{aligned}$$

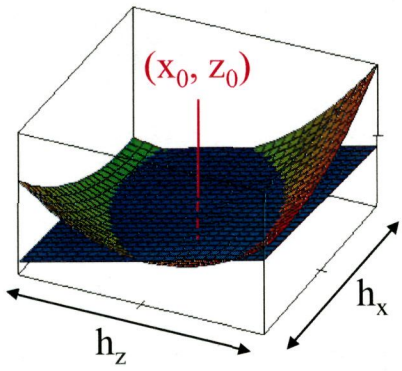


Figura C.7: Paraboloide y plano equivalente

Por tanto, igualando las expresiones de los volúmenes de ambas superficies

$$\int_{z_0-h_z/2}^{z_0+h_z/2}\int_{x_0-h_x/2}^{x_0+h_x/2}\left(a+bx+cz+dx^2+fxz+gz^2\right)dxdz=\int_{z_0-h_z/2}^{z_0+h_z/2}\int_{x_0-h_x/2}^{x_0+h_x/2}\left(a'+b'x+c'z\right)dxdz$$

obtenemos la expresión para el coeficiente a':

$$a'=a+\frac{1}{12}\left(dh_x^2+gh_z^2\right)-\left(dx_0^2+fx_0z_0+gz_0^2\right)$$

El valor realmente interesante de esta propiedad es el desplazamiento vertical sobre el punto (x_0, z_0) entre ambas superficies, D, cuya expresión es

$$D=\frac{1}{12}dh_x^2+\frac{1}{12}gh_z^2$$

Esta medida está íntimamente relacionada con la curvatura del paraboloide en dicho punto. Cuanto más pequeña sea, más cerca estará el plano equivalente del paraboloide, y por tanto, teniendo en cuenta que sus volúmenes serán iguales, más similares serán ambas superficies. Por otro lado, la utilidad de esta propiedad radica también en que, si el paraboloide no toca ni el suelo ni el techo del prisma, podemos usar la función **VolumenVoxelPlano()** aplicada al plano equivalente para obtener el volumen del paraboloide.

C.2.2 Función recursiva para el cálculo del volumen

El pseudo-código de la función para calcular el volumen del paraboloide sería así:

Funcion VolumenVoxelParab (xl, xr, yd, yu, zb, zf, a, b, c, d, f, g)

```
x0=(xl+xr)/2; z0=(zb+zf)/2
hx=xr-xl; hy=yu-yd; hz=zf-zb
D = (d*hx*hx+g*hz*hz) / 12
Calcular la altura del paraboloide sobre las 4 esquinas (ylb, ylf, yrb, yrf)
// casos directos.....................................................
Si las 4 esquinas estan por encima del prisma Retornar (1)
```

```
Si las 4 esquinas estan por debajo del prisma Retornar (0)
Si las 4 esquinas estan en el interior o D<umb1 o hy<umb2 Entonces
   Calcular plano equivalente (a', b', c')
   Retornar (VolumenVoxelPlano (xl, xr, yd, yu, zb, zf, a', b', c')
Fin Si
// subdividir el prima en 8 subprismas.....................
vol += VolumenVoxelParab (xl, x0, yd, y0, zb, z0, a, b, c, d, f, g)
vol += VolumenVoxelParab (xl, x0, yd, y0, z0, zf, a, b, c, d, f, g)
vol += VolumenVoxelParab (xl, x0, y0, yu, zb, z0, a, b, c, d, f, g)
vol += VolumenVoxelParab (xl, x0, y0, yu, z0, zf, a, b, c, d, f, g)
vol += VolumenVoxelParab (x0, xr, yd, y0, zb, z0, a, b, c, d, f, g)
vol += VolumenVoxelParab (x0, xr, yd, y0, z0, zf, a, b, c, d, f, g)
vol += VolumenVoxelParab (x0, xr, y0, yu, zb, z0, a, b, c, d, f, g)
vol += VolumenVoxelParab (x0, xr, y0, yu, z0, zf, a, b, c, d, f, g)
v = (xr-xl) * (yu-yd) * (zf-zb)
Retornar (vol/8/v)
```

Los valores *umb1* y *umb2* son los umbrales para la distancia entre el plano equivalente y el tamaño mínimo del prisma. En las pruebas con imágenes hemos usado valores de 0.01 para la distancia vertical y 0.1 para el tamaño mínimo (ambos valores multiplicados por el tamaño de un voxel).

C.3 Volumen bajo un paraboloide rotado interior a un prisma

Un paraboloide rotado viene dado por la expresión

$$A + Bx + Cy + Dz + Exy + Fxz + Gyz + Hx^2 + Iy^2 + Jz^2 = 0$$

La integración de esta superficie sobre un recinto rectangular del plano xz es demasiado compleja como para resolverla analíticamente. Por tanto, vamos a plantear un esquema recursivo como el de la sección anterior. La idea es la siguiente: inicialmente calcularemos el paraboloide vertical que comparte posición, orientación y curvaturas en el punto intersección con la vertical central del prisma, (x_0, y_0, z_0) (aplicando el lema 10.2). Si ambos paraboloides son similares, simplemente haremos uso de la función **VolumenVoxelParab()** definida en la sección anterior. En caso contrario, subdividiremos el prisma en 8, y volveremos a plantear el esquema a cada subprisma.

Para medir la similitud entre ambos paraboloides podemos mirar las distancias en vertical entre uno y otro sobre las 4 esquinas del prisma. Si la suma de las distancias se encuentra por debajo de un umbral, y teniendo en cuenta que ambos paraboloides comparten posición, orientación y curvaturas en el punto central, podemos considerar que el volumen bajo ambas superficies será muy parecido. Por ejemplo, en la figura C.8 se muestra la gráfica de un paraboloide rotado, junto con dos de sus paraboloides verticales (suponiendo que ya se ha subdividido el prisma inicial). Podemos ver que el paraboloide vertical de la izquierda es bastante similar, con lo cual su volumen será prácticamente el mismo que el de esa sección del paraboloide rotado. Sin embargo, atendiendo al paraboloide vertical de la derecha, en función de la magnitud de la distancia D habrá que considerar si volver a subdividir o usar el volumen de dicho paraboloide como estimación.

El pseudo-código de la función sería el siguiente:

Funcion VolumenVoxelParabRot (xl, xr, yd, yu, zb, zf, A, B, C, D, E, F, G, H, I, J)

```
x0=(xl+xr)/2; y0=(yd+yu)/2; z0=(zb+zf)/2
Calcular el paraboloide vertical con el lema 10.2 (a, b, c, d, f, g)
```

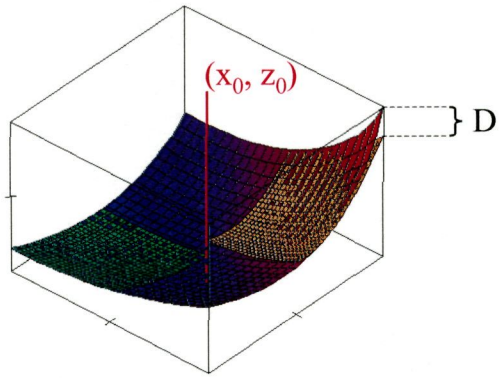


Figura C.8: Paraboloide rotado y paraboloides verticales similares en distintas zonas

```
Calcular la altura de ambos paraboloide sobre las 4 esquinas
// casos directos.....................................................
Si las 4 esquinas estan por encima del prisma Retornar (1)
Si las 4 esquinas estan por debajo del prisma Retornar (0)
Si la diferencia entre las 4 esquinas de ambos paraboloides es corta Entonces
   Retornar (VolumenVoxelParab (xl, xr, yd, yu, zb, zf, a, b, c, d, f, g))
Fin Si
// subdividir el prima en 8 subprismas..............................
vol += VolumenVoxelParabRot (xl, x0, yd, y0, zb, z0, A, B, C, D, E, F, G, H, I, J)
vol += VolumenVoxelParabRot (xl, x0, yd, y0, z0, zf, A, B, C, D, E, F, G, H, I, J)
vol += VolumenVoxelParabRot (xl, x0, y0, yu, zb, z0, A, B, C, D, E, F, G, H, I, J)
vol += VolumenVoxelParabRot (xl, x0, y0, yu, z0, zf, A, B, C, D, E, F, G, H, I, J)
vol += VolumenVoxelParabRot (x0, xr, yd, y0, zb, z0, A, B, C, D, E, F, G, H, I, J)
vol += VolumenVoxelParabRot (x0, xr, yd, y0, z0, zf, A, B, C, D, E, F, G, H, I, J)
vol += VolumenVoxelParabRot (x0, xr, y0, yu, zb, z0, A, B, C, D, E, F, G, H, I, J)
vol += VolumenVoxelParabRot (x0, xr, y0, yu, z0, zf, A, B, C, D, E, F, G, H, I, J)
v = (xr-xl) * (yu-yd) * (zf-zb)
Retornar (vol/8/v)
```

Referencias bibliográficas

[ACT94] S. T. Acton, A. C. Bovik, and M. M. Crawford. Anisotropic Diffusion Pyramids for Image Segmentation. Proc. First IEEE Int. Conf. Image Processing, 3, pp. 478-482, 1994

[ALM00] A. Almansa and T. Lindeberg. Fingerprint Enhancement by Shape-adaptation of Scale-space Operators with Automatic Scale Selection. IEEE Trans. on Image Processing, 9(12), pp. 2027-2042, 2000

[ALV92] L. Alvarez, P. L. Lions, and J. M. Morel. Image Selective Smoothing and Edge Detection by Nonlinear Diffusion. SIAM J. Numer. Anal. 29, pp. 845-866, 1992

[ALV93] L. Alvarez, F. Guichard, P. L. Lions and J. M. Morel. Axioms and Fundamental Equations of Image Processing. Archive for Rational Mechanics and Analysis, 123(3), pp. 199-257, 1993

[ALV01] L. Alvarez, K. Krissian and A. Trujillo. Invariant Computation of Differential Characteristics in 3D Image. Cuadernos del Instituto Universitario de Ciencias y Tecnologías Cibernéticas, Universidad de Las Palmas, Num 13, 2001

[AST02] J. A. D. Aston, V. J. Cunningham, M-C. Asselin, A. Hammers, A. C. Evans and R. N. Gunn. Positron Emission Tomography Partial Volume Correction: Estimation and Algorithms. J. Cereb. Blood Flow Metab., 22, pp. 1019-1034, 2002

[AUB02] G. Aubert and P. Kornprobst. Mathematical Problems in Image Processing. Springer, 2002

[APO66] T. M. Apostol. Calculus, 2nd Edition. Xerox College Publishing, Waltham. 1966

[BEN75] J. R. Bennet and J. S. McDonald. On the Measurement of Curvature in a Quantised Environment. IEEE Trans. on Computers, C-24(8), pp803-820, 1975

[BHA96] P. Bhattacharya and D. Wild. A new Edge Detector for Gray Volumetric Data. Comput. Biol. Med., 26, pp. 315-328, 1996

[BIG91] J. Bigün. Multidimensional Orientation Estimation with Applications to Texture Analysis and Optical Flow. IEEE Trans. on Pat. Anal. and Mach. Intel., 13(8), pp. 775-790, 1991

[BOM90] M. Bomans, K. Hohne, U. Tiede and M. Riemer. 3D Segmentation of MR Images of the Head for 3D Display. IEEE Trans. on Medical Imaging, 9(2), pp. 177-183, 1990

[BRE00] M. Brejl and M. Sonka. Directional 3D Edge Detection in Anisotropic Data: Detector Design and Performance Assessment. Computer Vision and Image Understanding, special issue on Analysis of Volumetric Images, pp. 84-110, 2000

[BUH04] K. Buhler, P. Felkel and A. La Cruz. Geometric Methods for Vessel Visualization and Quantification - A Survey. In Geometric Modeling for Scientist Visualization. Heidelberg, Springe, pp. 399-420, 2004

[CAN83] J. F. Canny. Finding edges and lines in images. Master's thesis, MIT. AI Lab. TR-720, 1983.

[CAN86] J. F. Canny. A computational approach to edge detection. IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-698, 1986.

[CAS79] K. R. Castleman, R. H. Selzer and D. H. Blankenhorn. Edge Detection in Angiograms - An Application of the Wiener Filter. J.K. Aggarwal, ed., Digital Signal Processing, Point Lobos Press, N. Hollywood, CA, 1979.

[CAT92] F. Catte, P. L. Linos, J. M. Morel, and T. Coll. Image Selective Smoothing and Edge Detection by Nonlinear Diffusion. SIAM J. Num. Anal. 29, pp182-193, 1992

[CHA00] C. R. Chan, W. C. Karl and R. S. Lees. A New Model Based Technique for Enhanced Small Vessel Measurements in X-ray Cineangiograms. IEEE Trans. Med. Imag. 19(3), pp. 243-255, 2000

[CHR94] H. I. Christensen and J. L. Crowley. Experimental Environments for Computer Vision and Image Processing. Series in Machine Perception Artificial Intelligence, 11. World Scientific. 1994

[CLA93] T. A. Clarke, M. A. R. Cooper and J. G. Fryer. An Estimator for the Random Error in Subpixel Target Location and its use in the Bundle Adjustment. Optical 3-D Measurements Techniques II, Pub. Wichmann, pp. 161-168, 1993

[COS92] D. Cosandier and M. A. Chapman. High Precision Target Location for Industrila Metrology. Videometrics, Proc. SPIE 1820, pp. 111-122, 1992

[COT93] G. H. Cottet and L. Germain. Image Processing through Reaction combined with Nonlinear Diffusion. Math. Comp., 61, pp. 659-673, 1993

[DAN01] P. Danielsson and Q. Lin. Efficient Detection of Second-Degree Variations in 2D and 3D Images. Journal of Visual Communication and Image Repres., 12, pp. 255-305, 2001

[DEM95] D. Demigny, F. G. Lorca and L. Kessal. Evaluation of Edgte Detectors Performances with a Discrete Expression of Canny´s Criteria. IEEE International Conference on Image Processing, pp. 169-172, 1995

[DER87] R. Deriche. Using Canny's criteria to derive an optimal edge detector recursively implemented. The International Journal of Computer Vision, 1:167-187, April 1987.

[DER90] R. Deriche. Fast algorithms for Low Level Vision. IEEE Trans. on Pat. Anal. and Mach. Int., 12(1), pp. 78-87, 1990

[EIC88] P. H. Eichel, E. J. Delp, K. Koral and A. J. Buda. A Method for a Fully Automatic Definition of Coronary Arterial Edges From Cineangiograms. IEEE Trans. Med. Imag. 7(4), pp. 313-320, 1988

[FLE92] M. Fleck. Multiple widths yield reliable finite differences. IEEE Transactions PAMI, 14(4): 412-429, April 1992.

[FRA98] A. Frangi, W. J. Niessen, K. L. Vincken and M. A. Viergever. Multiscale Vessel Enhancement Filtering. Lecture Notes in computer Science, 1496, pp. 139-137, 1998

[FUK80] T. Fukui, M. Yachida and S. Tsuji. Detection and Tracking of Blood Vessels in Cineangiograms. Proc. of Iter. Conf. on Pattern Recognition, pp. 383-385, 1980

[GEI91] D. Geiger and A. Yuille. A Common Framework for Image Segmentation. Int. J. Comput. Vision, 6, pp. 227-243, 1991

[GER92] G. Gerig, O. Kubler, R. Kikinis and F. A. Jolesz. Nonlinear Anisotropic Filtering of MRI data. IEEE Trans. on Medical Imaging, 11(2), pp. 221-232, 1992

[GOE70] A. Goetz. Introduction to Differential Geometry. Addison-Wesley, Reading, MA USA. 1970

[GON92] H. Gongzhu and C. Xingping. A Subpixel Edge Detector using Expectation of First-order Derivatives. Image Processing Algorithms and Techniques III, Proc. SPIE 1657, pp. 415-425, 1992

[GRO78] F. Groan and P. Verbeek. Freeman-Code Probabilities of Object Boundary Quantized Contours. Computer Vision, Graphics, Image Processing, 7, pp. 391-402, 1978

[HAM95] C. Hamitouche, C. Roux and J. L. Coatrieux. Design of new Surface Detection Operators in the case of an Anisotropic Sampling of 3D Volume Data. Lecture Notes in Computer Science, Computer Vision, Virtual Reality & Robotic in Medicine, pp. 523-532, 1995

[HAR88] C. Harris and M. Stephens. A Combined Corner and Edge Detector. Proc. Fourth Alvey Vision Conference, pp.147-151, 1988

[HAR92] R. M. Haralick and L. G. Shapiro. Computer and Robot Vision, Vol.1. Addison-Wesley, Reading, MA, 1992

[HEA97] D. Hearn and M. P. Baker. Computer Graphics, 2nd Edition. Prentices-Hall International, Inc. 1997

[HEA97b] M. D. Heath, s. Sarkar, T. Sanocki and K. W. Bowyer. A Robust Visual Method of Assessing the Relative Performance of Edge Detection Algorithms. IEEE Trans. on Pat. Anal. and Mach. Intel., 19(12), pp. 1338-1359, 1997

[HWA02] S. H. Hwang and F. W. Wehrli. Subvoxel Processing: a Method for Reducing Partial Volume Blurring with Application to In Vivo MR Images of Trabecular Bone. Magnetic Resonance in Medicine, 47, pp. 948-957, 2002

[IBA96] L. Ibáñez, C. Hamitouche and C. Roux. Moment-Based Operator for Sub-Voxel Surface Extraction in Medical Imaging. Proc. IEEE International Conference on Image Processing (ICIP'96), pp. 277-280, 1996

[ITO95] T. Itoh and K. Koyamada. Automatic Isosurface Propagation Using an Extrema Graph and Sorted Boundary Cell Lists. IEEE Trans. Visual. Comput. Graphics, 1(4), pp. 319-327, 1995

[JAH95] B. Jahne. Digital Image Processing: Concepts, Algorithms, and Scientific Applications, 3rd Edition. Springer-Verlag. 1995

[JAI89] A. K. Jain. Fundamentals of Digital Image Processing. Prentice-Hall International Editions. 1989

[JAI95] R. Jain, R. Kasturi and B. B. Schunck. Machine Vision. McGraw-Hill, New York, 1995

[KAY93] T. Kayikcioglu and S. Mitra. A Novel Method for Three Dimensional Reconstruction of Coronary Arterial Trees from Biplane Angiograms. Proc. of SPIE Med. Imag. VII, pp. 62-73, 1993

[KAY02] T. Kayikcioglu, A. Gangal, M. Turhal and C. Kose, A Surface-based Method for Detection of Coronary Vessel Boundaries in poor quality X-ray Angiogram Images, Pattern Recognition Letters, 23(7), pp. 783-802, 2002

[KHO97] Khoral Research Inc. Khoros Professional Student Edition. Addison-Wesley Pub Co, 1997

[KIR82] R. L. Kirkeeide, P. Fung, R. W. Smalling and L. Gould. Automated Evaluation of Vessel Diameter from Arteriograms. IEEE Trans. on Comput. Cardiol, pp. 215-218, 1982

[KIR03] C. Kirbas and F. Quek. A Review of Vessel Extraction Techniques and Algorithms. Tech. Rep. VisLab Wright State University, Dayton, Ohio, 2000

[KIS94] M. Kisworo, S. Venkatesh and G. West. Modeling Edges at Subpixel Accuracy using the Local Energy Approach. IEEE Trans. on Pat. Anal. and Mach. Intel., 16(4), pp. 405-410, 1994

[KIT82] L. Kitchen and A. Rosenfeld. Gray-Level Corner Detection, Pattern Recog. Lett., 1(2), pp95-102, 1982

[KIT88] K. Kitamura, J. M. Tobis and J. Sklansky. Estimating the 3D Skeletons and Transverse Areas of Coronary Arteries from Biplane Angiograms. IEEE Trans. Med. Imag. 7(3), pp. 173-187, 1988

[KLE97] A. K. Klein, F. Lee and A. A. Amini. Quantitative Coronary Angiography with Deformable Spline Models. IEEE Trans. Med. Imag., 16(5), pp. 468-482, 1997

[KOL95] T. M. Koller, G. Gerig, G. Szekely and D. Dettwiler. Multiscale Detection of Curvilinear Structures in 2D and 3D Image Data. Proc. of Int. Conf. on Computer Vision (ICCV'95), pp. 864-869, 1995

[KON94] K. Konstantinides and J. Rasure. The Khoros Software Development Environment for Image and Signal Processing. IEEE Journal of Image Processing, 3(3), pp. 243-252, 1994

[KOO82] C. J. Kooijman, H. H. C. Reiber, J. J. Gerbrands, J. C. H. Schuurbiers, C. J. Slager, A. D. Boer and P. W. Serruys. Computer-aided Quantitation of the Severity of Coronary Obstructions from Single View Cineangiograms. Proc. of IEEE Comp. Soc. Internat. Symp. on Med. Imag. Image Interpretation, pp. 59-64, 1982

[KRI97] K. Krissian, G. Malandain and N. Ayache. Directional Anisotropic Diffusion Applied to Segmentation of Vessels in 3D images. Scale-Space Theory in Comp. Vision. Lecture Notes in Computer Science, Springer Verlag, pp. 345-348, 1997

[KRI00a] K. Krissian. Multiscale Analysis: Applications to Medical Imaging and 3D Vessel Detection. PhD Thesis. University of Nice-Sophia Antipolis, 2000.

[KRI00b] K. Krissian, R. Vaillant, Y. Trousset, G. Malandain and N. Ayache. Automatic and Accurate Measurement of a Cross-sectional Area of Vessels in 3D X-ray Angiography Images. Proc. of SPIE'medical imaging 2000, Image Processing Conference, 2000

[KRI00c] K. Krissian, G. Malandain, N. Ayache, R. Vaillant and Y. Trousset. Model based Detection of Tubular Structures in 3D Images. Computer Vision and Image Understanding, 80:2, pp. 130-171, 2000

[KRI00d] K. Krissian. Flux-based Anisotropic Diffusion: Application to Enhancement of 3D Angiographies. Cuadernos del Instituto Universitario de Ciencias y Tecnologías Cibernéticas, Universidad de Las Palmas, Num 11, 2000

[KRI02] K. Krissian. Flux-based Anisotropic Diffusion: Application to Enhancement of 3D Angiogram. IEEE Trans. on Medical Imaging, 22(11), pp. 1440-1442, 2002

[LEE93] C. K. Lee, M. Haralick and K. Deguchi. Estimation of Curvature from Sampled Noisy Data, ICVPR'93, pp. 536-541, 1993

[LI88] Y. S. Li, T. Y. Young and J. A. Magerl. Subpixel Edge Detection and Estimation with a Microprocessor-Controlled Line Scan Camera". IEEE Trans. Ind. Elec., 35(1), pp. 105-112, 1988

[LIN94] T. Lindeberg. Scale-Space Theory in Computer Vision. Kluwer Academic Publishers. 1994

[LIV96] Y. Livnat, H. W. Shen and C. Johnson. A Near Optimal Isosurface Extraction Algorithm using the Span Space". IEEE Trans. Visual. Comput. Graphics, 2(1), pp. 73-84, 1996

[LOH98] G. Lohmann. Volumetric Image Analysis. Wiley Teubner Publishers, 1998

[LOR87] W. E. Lorensen and H. E. Cline. Marching Cubes: a high Resolution 3D Surface Reconstruction Algorithm. Computer Graphics, 21(4), 1987

[LOR97] C. Lorenz, I. C. Carsen, T. M. Buzug, C. Fassnacht and J. Weese. Multi-scale Line Segmentation with Automatic Estimation of Width, Contrast and Tangential Direction in 2D and 3D Medical Images. Lecture Notes in Computer Science, 1205, pp. 213-222, Springer Verlag, 1997

[LUO93] L. M. Luo, C. Hamitouche, L. Dillenseger and J. L. Coatrieux. A Moment-Based Three-Dimensional Edge Operator. IEEE Trans. on Biomedical Engineering, 40(7), pp. 693-703, 1993

[LUO94] D. S. Luo, M. A. King and S. Glick. Local Geometry Variable Conductance Diffusion for Post-reconstruction Filtering. IEEE Trans. on Nuclear Sciences, 41, pp. 2800-2806, 1994

[MAR80] D. Marr and E. C. Hildreth. Theory of edge detection. Proceedings of the Royal Society, London B, 207:187-217, 1980

[MAR82] D. Marr. Vision. Freeman, 1982.

[MEH94] R. Mehrotra and S. Zhan. A Zero-crossing-based Optimal Three-dimensional Edge Detector. CVGIP Image Understanding, 59, pp. 242-253, 1994

[MON91] O. Monga, R. Deriche and J. M. Rocchisani. 3D Edge Detection using Recursive Filtering. Computer Vision, Graphics and Image Processing, 53(1), pp. 76-87, 1991

[MON95] O. Monga and S. Benayoun. Using Partial Derivatives of 3D Images to Extract Typical Surface Features. Computer Vision and Image Understanding, Vol. 61(2), pp. 171-189, 1995

[NGU86] T. V. Nguyen and J. Sklansky. Computing the Skeleton of Coronary Arteries in Cineangiograms. Comput. Biomed. Res. 19, pp. 428-444, 1986

[NIE91] G. M. Nielson and B. Hamann. The Asymptotic Decider: Resolving the Ambiguity in Marching Cubes. Proc. IEEE Visualization, pp. 83-90, 1991

[NIK01] N. Nikolaidis and I. Pitas. 3-D Image Processing Algorithms. John Wiley & Sons, Inc., 2001

[NIX02] M. Nixon and A. Aguado. Feature Extraction and Image Processing. Newnes, Butterworth-Heinemann. 2002

[PAP88] T. N. Pappas and J. S. Lim. A Method for Estimation of Coronary Artery Dimensions Angiograms. IEEE Trans. Acoust. Speech Signal Processing, 36(9), pp. 1501-1513, 1988

[PAR97] J. R. Parker. Algorithms for Image Processing and Computer Vision. Wiley Computer Publishing, 1997

[PER90] P. Perona and J. Malik. Scale-space and Edge Detection using Anisotropic Diffusion. IEEE Trans. Pattern Anal. Mach. Intell. 12, pp. 629-639 (1990)

[PIT90] I. Pitas and A. N. Venetsanopoulos. Nonlinear Digital Filters: Principles and Applications. Kluwer, 1990

[POL97] R. Polli and G. Valli. An Algorithm for Real-time Vessel Enhancement and Detection. Comput. Meth. Programs Biomed. 52, pp. 1-22, 1997

[PRA01] W. K. Pratt. Digital Image Processing, 3rd Edition. John Wiley & Sons, Inc. 2001

[PRE66] J. M. S. Prewitt and M. L. Mendelsohn. The Analysis of Cell Images, Ann. N. Y. Acad. Sci., 128, pp. 1035-1053, 1966

[REI84] J. H. C. Reiber, C. J. Kooijman, C. J. Slager, J. J. Gerbrands, J. C. H. Schuurbiers, A. D. Boer, W. D. Boer, W. Wijns, P. W. Serruys and G. Hugenholtz. Coronary Artery Dimensions from Cineangiograms-methodology and Validation of a Computer-assisted Analysis Procedure. IEEE Trans. Med. Imag. MI-3, pp. 131-141, 1984

[RIF99] H. Rifai, I. Bloch, S. A. Hutchinson, J. Wiart, and L. Garnero. Segmentation of the Skull in MRI Volumes Using Deformable Models and Taking the Partial Volume Effect into Account. Proc. of SPIE Medical Imaging Symposium, San Diego, pp. 288-299, 1999

[ROB65] L. G. Roberts. Machine perception of three-dimensional solid. Optical and Electrooptical Information Processing. MIT Press, Cambridge, MA. pp. 159-167, 1965

[ROG89] D. F. Rogers and J. A. Adams. Mathematical Elements for Computer Graphics, 2nd Edition. McGraw-Hill International Edition, 1989

[ROM94] B. M. H. Romeny. Geometry-Driven Diffusion in Computer Vision. Kluwer Academic Publishers, 1994

[ROS82] A. Rosenfeld and A. C. Kat. Digital Picture Processing, 2nd Edition. Academic Press, NY, 1982

[SAN95] P. Santago and H. D. Gage. Statistical Models of Partial Volume Effect. IEEE Trans. on Image Processing, 4(11), pp. 1531-1540, 1995

[SAN99] G. I. Sánchez-Ortiz, D. Rueckert and P. Burger. Knowledge-based Anisotropic Diffusion of Cardiac Magnetic Resonance Images. Medical Image Analysis, 3(1), pp. 77-101, 1999

[SAP01] G. Sapiro. Geometric Partial Differential Equations and Image Analysys.Cambridge University Press, 2001

[SAT97] Y. Sato, S. Nakajima, H. Atsumi, T. Koller, G. Gerig, S. Yoshida and R. Kikinis. 3D Mulsti-scale Line Filter for Segmentation and Visualization of Curvilinear Structures in Medical Images. Lecture Notes in Computer Science, 1205, pp. 213-222, Springer Verlag, 1997

[SAT98] Y. Sato, S. Nakajima, N. Shiraga, H. Atsumi, S. Yoshida, T. Koller, G. Gerig and R. Kikinis. Three-dimensional Mulsti-scale Line Filter for Segmentation and Visualization of Curvilinear Structures in Medical Images. Medical Image Analysis, 2(2), pp. 143-168, 1998

[SEB97] J. M. Sebastián-Zúñiga, O. Reinoso, R. Aracil, D. García and F. Torres. Reconstruction of Step Edges with Subpixel Accuracy in Graylevel Images. Proc. of SPIE Image Preconstruction and Restoration II, 3170, pp. 215-226, 1997

[SEI88] P. Seitz. Optical Super-resolution using Solid-state Cameras and Digital Signal Processing. Optical Engineering, 27(7), pp. 535-540, 1988

[SEN02] M. Sengupta, L. Hohri, C. C. Tsao, M. Thompson and T. Lundquist. Sub-resolution Placement using IR Image to CAD Database Alignment: an Algorithm for Silicon-side Probing. Image Processing: Algorithms and Systems, Proc. SPIE 4667, pp. 449-459, 2002

[SHE92] J. Shen and S. Castan. An Optimal Linear Operator for Step Edge Detection. Computer Vision, Braphics, and Image Processing: Graphical Models and Understanding, 54(2), pp. 112-133. 1992

[SHU83] K. Shumeli, W. R. Brody and A. Macovski. Estimation of Blood Vessel Boundaries in X-ray Images. Opt. Eng. 22(1), pp. 110-116, 1983

[SNY95] W. Snyder, Y. S. Han, G. Bilbro, R. Whitaker and S. Pizer. Image Relaxation: Restoration and Feature Extraction. IEEE Trans. Pattern Anal. Mach. Intell., 17, pp. 620-624, 1995

[SOB70] I. E. Sobel. Camera Models and Machine Perception, PhD Thesis, Stanford Univ., 1970

[SON95] M. Sonka, M. D. Winniford and S. M. Collins. Robust Simultaneous Detection of Coronary Borders in Complex Images. IEEE Trans. Med. Imag. 14, pp. 151-161, 1995

[SON98] M. Sonka. Image Processing, Analysis, and Machine Vision, 2nd Edition. PWS Publishing, 1998

[SON00] M. Sonka and J. M. Fitzpatrick. Handbook of Medical Imaging, Vol. 2 (Medical Image Processing and Analysis). Spie Press, 2000

[SPE83a] J. R. Spears, T. Sandor, A. V. Als, M. Malagold, J. E. Markis, V. Grossman, J. R. Serur and S. Paulin. Computerized Image Analysis for Quantitative Measurement of Vessel Diameter from Cineangiograms. Circulation 68(2), pp. 453-461, 1983

[SPE83b] J. R. Spears, T. Sandor, D. S. Balm and S. Paulin. The Minimum Error in Estimating Coronary Luminal Cross-sectional Area from Cineangiographic Diameter Measurements. Catherizat. Cardiovascular Diagnosis 9, pp. 119-128, 1983

[STE94] E. Steen and B. Olstad. Scale-space and Boundary Detection in Ultrasonic Imaging using Nonlinear Signal-adaptive Anisotropic Diffusion. Image Processing, SPIE Vol. 2167, pp. 116-127, 1994

[SUE02] P. Suetens. Fundamentals of Medical Imaging. Cambridge University Press, 2002

[SUM97] P. Summers, A. Bhalearo and D. Hawkes. Multiresolution, Model Based Segmentation of MRA. Journal of Magnetic Resonance Imaging, 7, pp. 950-957, 1997

[SUN89] Y. Sun. Automated Identification of Vessel Contours in Coronary Arteriograms by an Adaptive Tracking Algorithm. IEEE Trans. Med. Imag., 8(1), pp. 78-88, 1989

[SZE94] G. Szekely, T. M. Koller, R. Kikinis and G. Gerig. Structural Description and Combined 3D Display for Superior Analsys of Cerebral Vascularity from MRA. Visualization in Biomedical Computing, pp. 272-281, 1994

[TAB84] A. J. Tabatabai and O. R. Mitchell. Edge Location to Subpixel Values in Digital Imagery. IEEE Trans. in Pat. Anal. and Mach. Intel., 6(2), pp. 188-199, 1984

[TAN98] C. K. Tang and G. Medioni. Extremal Feature Extraction from 3-D Vector and Noisy Scalar Fields. Proc. IEEE Visualization, pp. 95-102, 1998

[TEB98] S. Teboul, L. Blanc-Féraud, G. Aubert and M. Barlaud. Variational Approach for Edge-preserving Regularization using Coupled PDE's. IEEE Trans. on Image Processing, 7(3), pp. 387-397, 1998

[THI95] J. P. Thirion and A. Gourdon. Computing the Differential Characteristic of Isointensity Surfaces. Computer Vision and Image Understanding, Vol. 61(2), pp. 190-202, 1995

[THI96] J. P. Thirion and A. Gourdon. The 3D Marching Lines Algorithm. Graph. Models Image Proc., 58(6), pp. 503-509, 1996

[TRU94] A. Trujillo and E. González. XMegaWave, un Entorno para Tratamiento de Imágenes. IX Simposium Nacional de la Unión Científica Internacional de Radio (URSI), 1994

[TRU98a] A. Trujillo and E. González. XMegaWave, an Image Processing Environment. IEEE Computer Vision and Pattern Recognition Conference (CVPR), 1998

[TRU98b] A. Trujillo and A. Medina. WinMegaWave, un Entorno para el Tratamiento de Imágenes en Windows. IV Jornadas Nacionales de Informática, 1998

[TRU01] A. Trujillo and K. Krissian. AMILab: a 2D/3D Image Processing Software. IEEE International Conference on Computer Vision, 2, pp. 744, 2001

[TRU01b] F. Truchetet, F. Nicolier and O. Laligant. Subpixel Edge Detection for Dimensional Control by Artificial Vision. Journal of Electronic Imaging, 10(1), pp. 234-239, 2001

[TSA94] D. M. Tsai and M. F. Chen. Curve Fitting Approach for Tangent Angle and Curvature Measurements. Pattern Recognition, 27(5), pp. 699-711, 1994

[VAN98] P. M. J. Van der Zwet, M. Nettesheim, J. J. Gerbrands and J. H. C. Reiber. Derivation of Optimal Filters for the Detection of Coronary Arteries. IEEE Trans. Med. Imag. 17(1), pp. 108-119, 1998

[VEN03] V. Venkatachalam and R. M. Wasswerman. Comprehensive Investigation of Subpixel Edge Detection Schemes in Metrology. Machine Vision Applications in Industrial Inspection. Proc. SPIE 5011, pp. 200-211, 2003

[VER96] B. Verdonck, L. Bloch, H. Maitre, D. Vandermeulen, P. Suetens and G. Marchal. Accurate Segmentation of Blood Vessels from 3D Medical Images. Proc. of IEEE Int. Conf on Image Processing, 3, pp. 311-314, 1996

[WAL94] R. J. Walkenburg, A. M. Mclvor and P. W. Power. An Evaluation of Subpixel Feature Localization Methods for Precision Measurements. Videometrics III, Proc. SPIE 2350, pp. 229-238, 1994

[WAT98] A. Watt and F. Policarpo. The Computer Image. Addison-Wesley, 1998

[WEB89] D. M. Weber. Absolute Diameter Measurements of Coronary Arteries based on the First Zero Crossing of the Fourier Spectrum. Med. Phys. 16(2), pp. 188-196, 1989

[WEI94a] J. Weickert. Anisotropic Diffusion Filters for Image Processing based Quality Control. Proc. Seventh European Conf. on Mathematics in Industry, pp. 355-362, 1994

[WEI94b] J. Weickert. Scale-space Properties of Nonlinear Diffusion Filtering with a Diffusion Tensor. Report 110, University of Kaiserslautern, Germany. 1994

[WEI98] J. Weickert. Anisotropic Diffusion in Image Processing. ECMI Series, Teubner, Stuttgart, 1998

[WES90] G. A. W. West and T. A. Clarke. A Survey and Examination of Subpixel Measurement Techniques". Close Range Photogrammetry Meets Machine Vision, Proc. SPIE 1395, pp. 456-463, 1990

[WHI93] R. T. Whitaker. Geometry limited Diffusion in the Characterization of Geometric Patches in Images. CVGIP: Image Understanding, 57, pp. 111.-120, 1993

[YIM00] P. J. Yim, R. Mullick, R. M. Summers, H. Marcos, J. R. Cebral, R. Lohner and P. L. Choyke. Measurement of Stenosis from Magnetic Resonance Angiography using Vessel Skeletons". Proc. SPIE Medical Imaging, Vol 3978, pp. 245-255, 2000

[YOO93] T. S. Yoo and J. M. Coggins. Using Statistical Pattern Recognition Techniques to Control Variable Conductance Diffusion. International Conference on Informtion Processing in Medical Imaging (IPMI), Lecture Notes in Computer Science, pp. 459-471, Springer, 1993

[YOU96] Y. L. You, W. Xu, A. Tannenbaum and M. Kaveh. Behavioral Analysis of Anisotropic Diffusion in Imgae Processing. IEEE Trans. Image Process. 5, pp1539-1553, 1996

[ZHA93] Y. J. Zhang. Quantitative Study of 3D Gradient Operators. Image and Vision Computing, 11(10), 1993

[ZUC81] S. W. Zucker and R. A. Hummel. A Three Dimensional Edge Operator. IEEE Trans. on Pat. Anal. and Mach. Intel., 3(3), pp. 324-331, 1981