UNIVERSIDAD POLITÉCNICA DE CARTAGENA

INGENIERÍA DE TELECOMUNICACIÓN

Proyecto Fin de Carrera

SEGMENTACIÓN DE SECUENCIAS DE IMÁGENES ESTEREOSCÓPICAS MEDIANTE COMPETICIÓN DE REGIONES PARA EL MODELADO 3D DE MÚSCULOS ARTIFICIALES

Santiago Fernando González Benítez

1 de febrero de 2005

Director: ***Rafael Verdú Monedero***

A Sergio y Fina

Agradecimientos

Este documento es el resultado final de muchos meses de trabajo. Durante este periodo de tiempo he tenido la oportunidad de adquirir una considerable cantidad de conocimientos acerca de la segmentación y el tratamiento digital de imágenes en general. Por ello, quisiera expresar mi agradecimiento a todas las personas que, directa o indirectamente, han contribuido a la elaboración de este proyecto. Más concretamente, quiero expresar mi gratitud a las siguientes personas.

A Rafael Verdú, en primer lugar, por concederme la oportunidad de trabajar en este proyecto. Por su dedicación e interés desde principio a fin, manteniendo siempre abierta la puerta de su despacho y dispuesto a ayudar en la medida de lo posible. Sobre todo, gracias por los conocimientos transmitidos y por contagiarme el gusto por el trabajo bien hecho.

A mis compañeros de promoción con los que he pasado tantas horas de clase, estudio, exámenes... Ha sido un placer compartir el camino con vosotros.

A mis padres y mis hermanos, por su apoyo e interés durante todo mi proceso de formación académica y en la elaboración de este proyecto, en particular.

Y muy muy especialmente, a Bárbara. La persona que ha sido mi principal apoyo durante todos mis años universitarios, incluyendo el periodo de elaboración de este proyecto en el que (como modelo) también ha contribuido.

Índice

Capítulo 1.

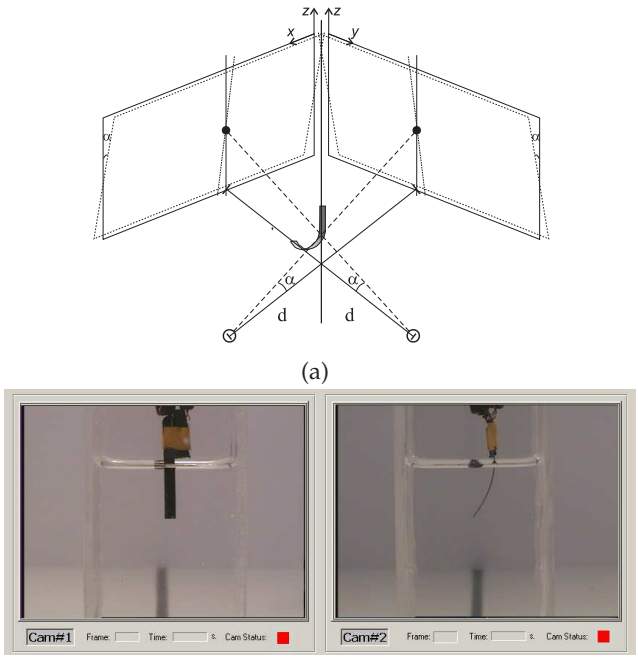
Introducción

En este proyecto se describe el estudio e implementación de un algoritmo para la segmentación de secuencias de imágenes estereoscópicas. Este algoritmo está pensado específicamente para la segmentación de imágenes de músculos artificiales. Los músculos artificiales son dispositivos construidos mediante polímeros conductores que son capaces de transformar energía eléctrica en trabajo mecánico mediante una reacción electroquímica. Estos dispositivos, sumergidos en agua y expuestos a una corriente eléctrica, son capaces de curvarse sobre si mismos y volver a enderezarse en función de dicha corriente.

Para la observación y caracterización mecánica de estos músculos se ha desarrollado un método basado en visión artificial y procesado de imagen. En una primera aproximación para abordar el problema, la adquisición de imágenes se realiza con una sola cámara y desde un plano perpendicular al movimiento [2]. Mediante la utilización de dos cámaras y el uso de técnicas de visión estereoscópica es posible la caracterización del músculo en 3D [3, 4].

Para dicha caracterización es fundamental el procesado digital de las imágenes obtenidas por las cámaras. Este procesado consiste en la segmentación del músculo y el seguimiento de éste a lo largo de su recorrido. Sin una segmentación correcta la extracción de los parámetros necesarios para caracterizar el músculo es imposible. Además, no podemos olvidar que tratamos de segmentar vídeo, esto es, secuencias de fotogramas que difieren ligeramente unos de otros. Esta circunstancia exige que nuestro algoritmo combine dos cualidades en principio opuestas: una gran robustez y la agilidad suficiente para seguir el movimiento del músculo. Por otra parte, aunque la literatura encontrada para la segmentación de imágenes es relativamente amplia no ocurre lo mismo con la segmentación y seguimiento de video ya que se trata de un problema difícil para el que, de momento, sólo existen soluciones parciales. La implementación del algoritmo que realiza este procesado digital de la imagen es el objetivo de este proyecto fin de carrera.

El punto de partida para la implementación de la solución final es el método de segmenta-



(a)

(b)

Figura 1.1.: Disposición de cámaras para la adquisición de imágenes (a) y fotogramas obtenidos (b).

ción *Competición de Regiones* de Zhu y Yuille [1]. Este algoritmo destaca frente a otros métodos por su gran robustez y sus ventajas frente a otros métodos serán explicadas más adelante.

1.1. Motivaciones

Los evolución de los materiales inteligentes, es decir, materiales capaces de responder de modo reversible y controlable ante diferentes estímulos físicos o químicos externos, ha sido espectacular en los últimos años. Dentro de estos materiales activos destacan los polímeros electroactivos por el interés que despiertan sus posibles aplicaciones. Una de las aplicaciones más interesantes de estos materiales son los llamados actuadores o *músculos artificiales* [5, 6]. Estos dispositivos, todavía en fase de investigación y desarrollo, podrían ser utilizados en multitud de ámbitos tan dispares como prótesis para extremidades humanas o la creación de sistemas motrices para vehículos de exploración de otros planetas.

Es precisamente esta perspectiva de futuro la que está impulsando a multitud de universidades de todo el mundo a desarrollar proyectos relacionados con este tema. El Centro de Electroquímica y Materiales Inteligentes de la UPCT tiene también una línea de investigación en este sentido [7]. Como hemos comentado anteriormente, se ha desarrollado un sistema de observación y caracterización de estos músculos mediante cámaras que captan el movimiento de las láminas

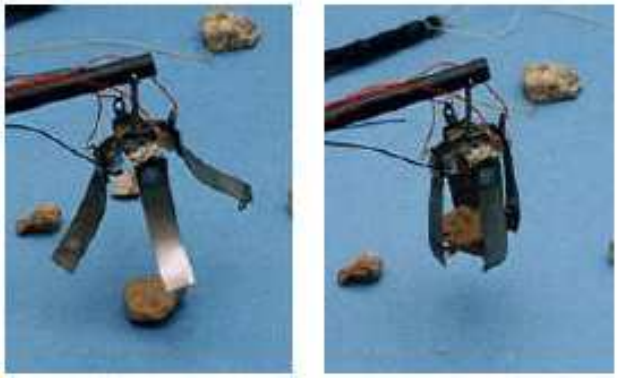


Figura 1.2.: Extremidad basada en músculos artificiales.

[8]. Para la caracterización de los músculos se sigue un método formado por partes bien diferenciadas que se corresponden a cada una de las etapas del proceso:

- Control de las cámaras y adquisición de imágenes estéreo.
- **Procesado de las imágenes**. Compuesto por dos tareas:
  - Segmentación de la imagen. Esta tarea se encarga de aislar el músculo del resto de la imagen.
  - Seguimiento del objeto.
- Extracción de parámetros: movimiento, energía de curvatura, etc.

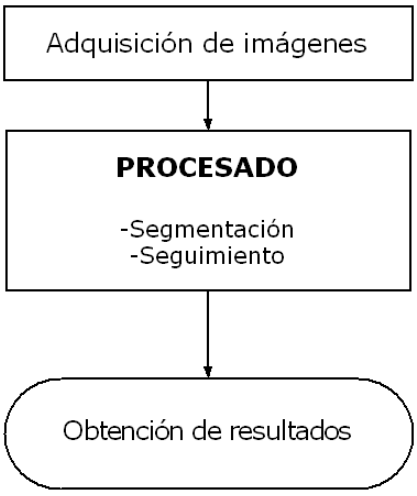


Figura 1.3.: Método para extracción de parámetros.

En esta memoria se describe el trabajo realizado para llevar a cabo el procesado de las imágenes, tanto de la segmentación como del seguimiento. La importancia de esta fase es evidente,

ya que una segmentación errónea o incapaz de seguir el movimiento del objeto provocaría la obtención de parámetros inexactos.

Además de la motivación práctica anteriormente expuesta, no es menos importante el estudio y desarrollo de un algoritmo de segmentación y seguimiento de un objeto en secuencias de imágenes. Si bien el algoritmo final está especializado en la segmentación de músculos artificiales, a lo largo del proceso de desarrollo del mismo se han realizado diversas pruebas con diferentes tipos de imágenes, tratando así de crear un método de segmentación válido para una gran variedad de imágenes. Durante el desarrollo de este proyecto se han estudiado varias técnicas de segmentación, especialmente *Competición de Regiones* de Zhu y Yuille [1], algoritmo que ha sido evolucionado y adaptado para la segmentación de vídeo estéreo.

1.2. Objetivos generales

El objetivo de este proyecto es el estudio de distintas técnicas y algoritmos de segmentación, valorando sus ventajas y desventajas en diferentes situaciones. En una segunda etapa, se realiza la implementación del algoritmo *Region Competition* (RC) propuesto por Zhu y Yuille [1]. Una vez implementado el algoritmo, se estudia su comportamiento con distintos tipos de imágenes y, por último, ha de servir para segmentar pares de imágenes estereoscópicas de músculos artificiales. El resultado obtenido debe ser útil para la obtención de parámetros de movimiento y energía de curvatura para cada instante de la secuencia de video.

1.3. Fases de desarrollo

Las fases por las que ha pasado el proceso de planteamiento y desarrollo del proyecto están resumidas en la siguiente lista y representadas en la figura 1.4.

1. Fase de documentación y experimentación práctica.
2. Análisis del problema.
3. Diseño inicial de la aplicación.
4. Implementación y desarrollo de la aplicación.
5. Evaluación de la aplicación y depuración de errores.
6. Presentación de resultados y conclusiones.

La fase de documentación comienza con la lectura comprensiva del artículo [1] y otros relacionados con el campo de la segmentación y procesado de imágenes. El estudio de técnicas básicas de segmentación mediante [25] fue de gran ayuda para la comprensión de los diferentes enfoques posibles ante el problema de la segmentación y para la redacción de esta memoria. En realidad, la fase de documentación se extiende a lo largo de toda la duración del proyecto. Paralelamente a la fase inicial de documentación se realizaron una serie de experimentos prácticos para conseguir cierta habilidad con las herramientas del software *Matlab* útiles para el procesado digital de imágenes.

El análisis del problema se solapa con la fase de documentación y con el inicio del diseño inicial de la aplicación ganando profundidad este análisis conforme se consideran alternativas de diseño para el algoritmo.

Los diseños iniciales de la aplicación son, básicamente, implementaciones simplificadas del algoritmo RC. En estos diseños iniciales todavía no se segmentan secuencias de video.

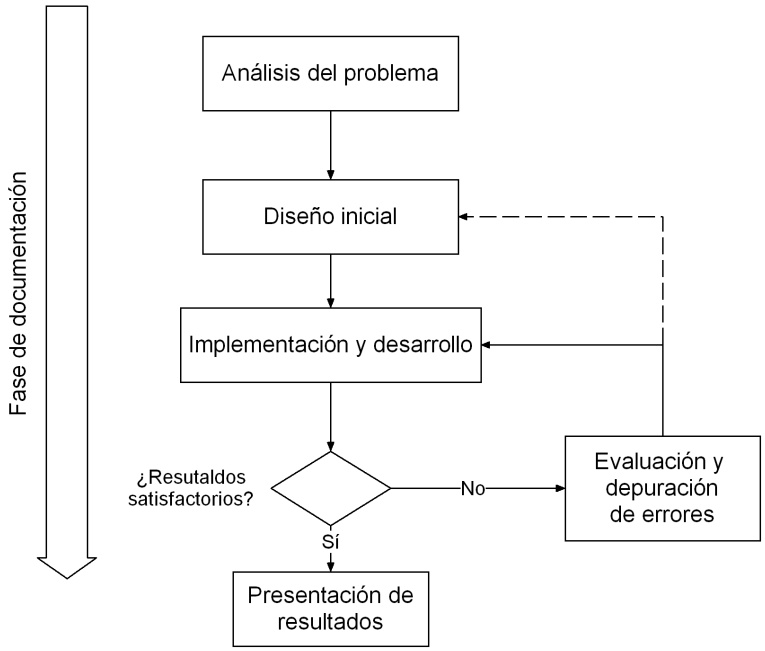


Figura 1.4.: Fases de desarrollo del proyecto.

La implementación de la aplicación final ha sido un desarrollo constructivo mediante el cual, nuestro algoritmo iba ganando funcionalidad y robustez paso a paso. Simultáneamente al desarrollo de la aplicación se va evaluando su funcionamiento según criterios de robustez (tratando de que la aplicación sea válida para la segmentación de un amplio rango de imágenes) y coste computacional (con objeto de optimizar su funcionamiento en la segmentación de secuencias estereoscópicas de músculos artificiales).

Finalmente, en la fase de presentación de resultados se refina la presentación de las imágenes segmentadas y la salida de la aplicación produce un vídeo segmentado de la secuencia de entrada. Esta fase de resultados y conclusiones engloba también el periodo de redacción de esta memoria y la defensa de la misma.

1.4. Organización de la memoria

Además de servir como memoria de todo el trabajo llevado a cabo durante la realización de este proyecto, en este documento también se estudian diferentes métodos de segmentación junto al finalmente elegido de forma que el lector encontrará también un amplio estudio de las diferentes soluciones posibles para el problema de segmentación de imágenes.

Esta memoria está organizada de la siguiente forma: Tras esta introducción se encuentra una primera parte donde se estudian a fondo las diversas técnicas de segmentación clásica, apuntando en muchos casos otras técnicas más sofisticadas que han sido derivadas de ellas. En esta primera parte, el lector encontrará un reflejo de todo el estudio previo a la realización de la aplicación de segmentación de vídeo mediante el algoritmo Competición de Regiones y además, una excelente guía sobre las diferentes aproximaciones al problema de segmentación de imágenes.

La siguiente parte está dedicada a la aplicación para segmentación de músculos artificiales mediante el algoritmo de Zhu y Yuille [1]. Primero se expone el algoritmo original y en las siguientes secciones se describe la optimización llevada a cabo para la segmentación de secuencias de fotogramas estereoscópicos.

En la última sección se encuentran las conclusiones y el análisis de los resultados obtenidos junto con las posibles líneas de trabajo futuro.

Capítulo 2.

Técnicas de segmentación

En esta parte de la memoria se estudian las diversas técnicas que forman el entorno de trabajo de este proyecto: la segmentación de imágenes. En primer lugar se introducen algunas definiciones básicas y más adelante se repasan diferentes técnicas básicas para la segmentación de imágenes.

El primer paso a la hora de extraer información de una imagen es, generalmente, la segmentación. El proceso mediante el cual se divide una imagen en regiones de características similares es lo que se llama *segmentar* una imagen.

El proceso de segmentación de una imagen puede ser abordado desde diferentes puntos de vista. Cada una de las estrategias posibles tiene sus ventajas e inconvenientes. Por lo tanto, la estrategia a elegir depende de las características de la imagen que queremos segmentar y de cuál es la información que queremos extraer de dicha segmentación. Por ejemplo, en una aplicación que se encargue de detectar los coches que pasan por un segmento de carretera lo importante es que sea capaz de distinguir los objetos de dimensiones correspondientes a un vehículo. En este caso, llevar el análisis de la imagen a un nivel de detalle mayor no tendría sentido. Teniendo en cuenta las características de la imagen y del objeto a segmentar ha de elegirse el enfoque más adecuado.

En las siguientes secciones se describen las estrategias básicas de segmentación más representativas. Los algoritmos de segmentación para imágenes monocromáticas se basan, generalmente, en una de las dos siguientes características básicas: discontinuidades o similitudes. En el caso de las discontinuidades, se intenta dividir la imagen basándose en cambios abruptos del nivel de gris. Los algoritmos basados en las similitudes se basan en el uso de umbrales y división, fusión y crecimiento de regiones. Ambas aproximaciones son útiles para segmentación de imágenes que varían en el tiempo (vídeo), si bien, en el segundo caso, el movimiento puede ser un factor muy útil para mejorar el rendimiento de los algoritmos de segmentación.

2.1. Definiciones básicas

En esta sección se definen conceptos básicos como la imagen digital y otros conceptos útiles en el contexto del procesado y segmentación de imágenes.

2.1.1. La imagen digital

Una imagen es una señal continua, es decir, está definida en un espacio continuo, sin embargo, la mayoría de los análisis son realizados sobre imágenes digitales. El proceso de digitalización de una imagen consiste en el paso del espacio continuo en que está definida a uno discreto, en particular del espacio $\mathbb{R}^2$ al espacio $\mathbb{Z}^2$ lo cual se consigue mediante la división de la imagen en pequeñas áreas. Esas áreas son denominadas píxeles (de la abreviatura de la denominación inglesa *pixel* de *picture element*). A cada píxel se le asigna el valor medio registrado dentro del área que representa.

Siempre vamos a considerar imágenes digitales discretas ya que los píxeles sólo van a tomar valores en un conjunto discreto. En el caso de las imágenes a niveles de gris este conjunto es $\{0, 1, \ldots, 255\}$ y en el caso de imágenes binarias los valores posibles están en el conjunto $\{0, 1\}$.

Más formalmente, una imagen binaria f es una aplicación de un subconjunto $\mathcal{D}$ de $\mathbb{Z}^2$ llamado el dominio de definición de f en el conjunto $\{0, 1\}$

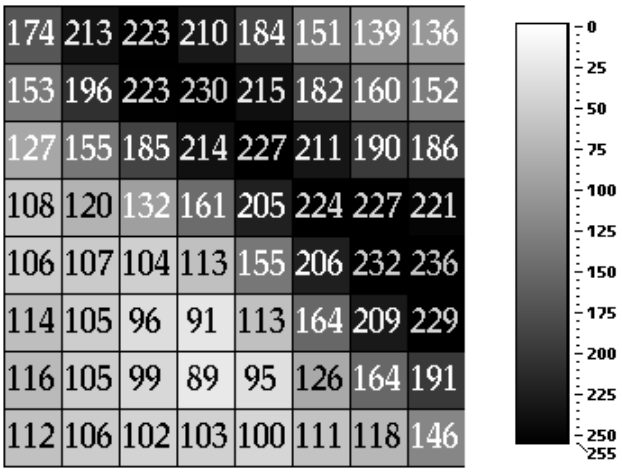
$$f : \mathcal{D} \subset \mathbb{Z}^2 \longrightarrow \{0, 1\} \ . \tag{2.1}$$

La definición formal de una imagen a niveles de gris sólo se diferencia de la anterior en que el conjunto en que toma valores la función está formado por más de dos enteros positivos. Más precisamente, una imagen f a niveles de gris es una aplicación de un subconjunto $\mathcal{D}$ en $\mathbb{Z}^2$, llamado dominio de definición de f, en un conjunto acotado o una secuencia de enteros no negativos:

$$f : \mathcal{D} \subset \mathbb{Z}^2 \longrightarrow N_0 = \{0, 1, \ldots, t_{max}\} \ , \tag{2.2}$$

donde t_{max} es el valor máximo alcanzable dependiendo del tipo de datos que se utilizan para almacenar la imagen (es decir, $2^n - 1$ si los píxeles tienen una capacidad de n bits). Las imágenes binarias son casos particulares de imágenes a niveles de gris. En la gran mayoría de los casos, el dominio siempre va a ser un subconjunto de $\mathbb{Z}^2$ y t_{max} igual a 1 ó bien a 255.

En la mayoría de los ejemplos considerados en esta memoria trataremos con imágenes a niveles de grises como la de la figura 2.1. Las imágenes digitales en color tienen una descripción



(a) (b)

Figura 2.1.: Ejemplo de imagen digital. (a) Imagen digital de 8×8 píxeles. (b) Escala de grises.

análoga pero, en lugar de usar un único número para cada píxel, usan tres. Uno para cada componente primario RGB (*red*, *green*, *blue*) que define el color del área representada por ese píxel. Dicho esto, las imágenes digitales en color pueden considerarse una extensión de las imágenes a niveles de grises y, desde el punto de vista del procesado digital pueden ser tratadas igualmente con un simple aumento de los parámetros del problema.

2.1.2. Procesado digital de imágenes. Segmentación.

Una vez definido el concepto de imagen digital asentaremos algunas definiciones básicas relativas al procesado digital de éstas y, más concretamente, a la segmentación.

Procesamiento de imágenes es el término usado para denominar las operaciones desarrolladas sobre un conjunto de datos de imagen para mejorarlas de alguna forma, para ayudar a su interpretación o para extraer algún tipo de información útil de ella. Es obvio que el procesamiento de imágenes no puede producir información a partir de nada. Si en el conjunto de datos no existe información concerniente a una aplicación o interpretación en particular, entonces no importa que cantidad de complicadas rutinas de procesamiento apliquemos, no se podrá obtener información.

Cuando se desea extraer información de una imagen, el primer paso suele ser la *segmentación*. La segmentación es el proceso mediante el cual se divide una imagen en regiones de características similares.

Procesado global y local

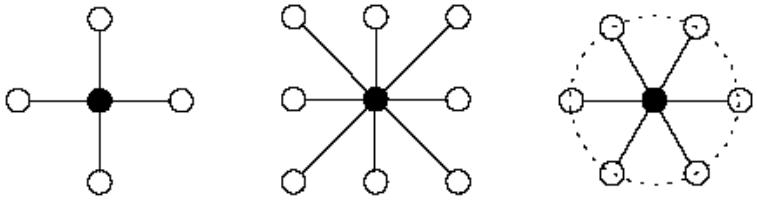
En algunas partes de esta memoria se distingue entre procesado *local* y *global*. La diferencia entre uno y otro puede explicarse fácilmente con un ejemplo: Si calculáramos la media de los valores de todos píxeles de la imagen de la figura 2.1 y después convirtiéramos esta imagen en binaria poniendo a 1 todos aquellos píxeles que superen el valor de esta media, estaríamos realizando un *procesado global* de la imagen. Si, alternativamente, realizásemos la misma operación por partes cogiendo grupos pequeños de píxeles para calcular su media y binarizar la imagen parte por parte estaríamos realizando un *procesado local* y la imagen binaria resultante sería distinta.

Bordes y discontinuidades

El término *borde* o *discontinuidad* de una imagen hace referencia a saltos abruptos en los valores de los píxeles presentes en ella. Por ejemplo, las imágenes de la figura 2.6 presentan discontinuidades al pasar de la zona clara a la oscura. Estos bordes tienen una alta probabilidad de estar relacionados con alguna parte de la imagen que puede resultar de interés y constituyen la base del funcionamiento de muchos algoritmos de segmentación.

Vecindad y relación de conectividad entre píxeles

La *vecindad* o vecinos de un píxel es el grupo de píxeles adyacentes a él que cumplen la *relación de conectividad* establecida en su contexto. En la figura 2.2 se ilustra más claramente este concepto.



(a) (b) (c)

Figura 2.2.: Diferentes tipos de conectividad de un píxel y su grupo de vecinos. (a) Conectividad a 4. (b) Conectividad a 8. (c) Conectividad a 6.

La figura 2.2(a) muestra un píxel y su grupo de vecinos en un entorno donde se ha definido una relación de conectividad a 4. La figura 2.2(b) es el caso análogo para una relación de conectividad a 8. La figura 2.2(c) ilustra el caso de conectividad a 6. Esta última es muy poco usada debido

a que casi no existen imágenes digitales que coloquen los píxeles de forma hexagonal y sólo está incluida aquí por sus excelentes propiedades de simetría desde un punto de vista algorítmico.

Objetos y fondo

En el contexto de la segmentación de imágenes es muy común clasificar los grupos de píxeles presentes en ella como *objetos* y *fondo*. Un *objeto* es un grupo de píxeles (generalmente conectados entre sí) que tiene cierto significado semántico dentro de la imagen y que, por lo tanto, suele ser el objetivo (*target*) de la segmentación. El fondo o *background* lo forman el resto de píxeles que no pertenecen a ningún objeto.

Regiones

Muchos algoritmos de segmentación se basan en el etiquetado de los píxeles de la imagen en función de su correlación con algún patrón o distribución probabilística que se espera coincida con el objeto a segmentar. Aquellos píxeles conectados entre sí y que tienen la misma etiqueta forman una *región*. Se espera que las regiones resultantes tras la segmentación se correspondan con *objetos* pero no tiene por qué ser así.

Descriptores regionales

Hemos comentado que el objetivo de la segmentación es dividir la imagen en regiones homogéneas de características similares. No obstante, hay que definir cuáles son las características de la imagen que nos interesa evaluar antes de efectuar la segmentación. Los *descriptores* son parámetros que nos dan información acerca de la imagen y en función de ellos decidiremos si un grupo de píxeles tienen características similares entre sí. El descriptor de una región puede ser, por ejemplo, la media de las intensidades de sus píxeles. En muchos tipos de imágenes el uso de un sólo descriptor puede ser suficiente para realizar la segmentación pero en otras ocasiones es necesario el uso de más descriptores y/o el uso de descriptores de orden superior. Por ejemplo, en una imagen formada por dos regiones con la misma media pero diferente varianza, el uso de la media como único descriptor sería inútil. Pueden implementarse algoritmos con multitud de descriptores como la media, la varianza, texturas, de tipo topológico o incluso basados en el dominio de la frecuencia. No obstante, dependiendo del algoritmo, el coste computacional puede resultar prohibitivo en algunos casos.

Filtrado digital

La naturaleza digital de las imágenes permite realizar procesos sobre ellas que de otra forma serían muy complicados o imposibles. Uno de los procesos más comunes sobre imágenes digitales es el *filtrado*. Hablando de forma general, puede considerarse un filtro cualquier operación que se realice sobre una imagen para obtener una versión modificada de ésta. Sin embargo, en el contexto del tratamiento digital de imágenes el *filtrado* hace referencia a operaciones más concretas.

Considerando la imagen digital como una señal, pueden aplicarse *filtros lineales* sobre ella. En la mayoría de los casos el objetivo del filtrado es el suavizado de la imagen por lo que es muy común el uso de filtros paso bajo. Este tipo de filtros también se usa para difuminar el efecto del ruido. En secciones posteriores se estudian diferentes formas de implementar filtros de este tipo (ver sección 2.3.2). No obstante, el uso de filtros lineales presenta dos inconvenientes: los bordes de la imagen se desdibujan y los picos causados por ruido impulsivo no pueden ser eliminados totalmente.

Una alternativa al filtrado lineal es el *filtrado morfológico* (ver el Anexo B). Mediante técnicas de morfología matemática pueden aplicarse filtros que sí son capaces de eliminar picos de ruido impulsivo además de otras características añadidas.

2.2. Técnicas basadas en detección de puntos y líneas

Aquí se presentan algunas técnicas básicas para detectar discontinuidades. Las discontinuidades más comunes en imágenes digitales son los *puntos*, las *líneas* y los *bordes*. En la práctica, la forma más usual de encontrar estas discontinuidades es recorrer la imagen mediante una máscara. Este procedimiento implica el cálculo de la suma de los productos de los coeficientes de la máscara y los valores de los niveles de grises de la imagen. Por ejemplo, para una máscara 3x3 como la de la figura 2.3 la respuesta será

$$R = w_1 z_1 + w_2 z_2 + ... + w_9 z_9 = \sum_{i=1}^{9} w_i z_i \ , \tag{2.3}$$

donde z_i es el nivel de gris del píxel asociado con el coeficiente de la máscara w_i.

Para la detección de puntos aislados se usará una máscara como la de la figura 2.4. De esta forma, lo que estamos haciendo es sopesar las diferencias entre el píxel central y sus vecinos. Diremos que el píxel evaluado en ese momento es un punto aislado cuando

$$|R| > T \ , \tag{2.4}$$

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

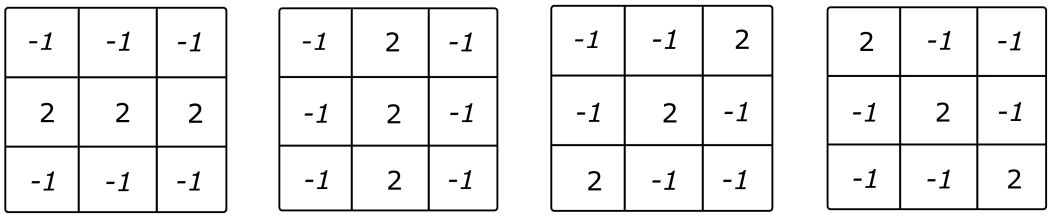
Figura 2.3.: Máscara general 3 x 3.

donde T es un umbral no negativo, y R viene dado por la ecuación 2.3.

-1	-1	-1
-1	8	-1
-1	-1	-1

Figura 2.4.: Máscara para detectar puntos aislados sobre un fondo uniforme.

Si lo que queremos es detectar líneas, las máscaras a usar serán diferentes dependiendo de la orientación de éstas. En la figura 2.5 se pueden ver ejemplos de máscaras para la detección de diferentes tipos de líneas. Dependiendo de los valores vecinos al píxel que estamos evaluando decidiremos si se trata de una linea o no. Usando diferentes máscaras diferentes máscaras podemos estimar la orientación de dicha línea.



(a) (b) (c) (d)

Figura 2.5.: Máscaras para detección de líneas. Detección de líneas horizontales (a), verticales (b) y oblicuas (c)(d).

Supongamos R_1, R_2, R_3 y R_4 las respuestas a las máscaras de la figura (de derecha a izquierda). Si aplicamos las cuatro máscaras a una imagen y, por ejemplo, $|R_1| > |R_j|$ para $j = 2, 3, 4$, diremos que ese punto en particular está más asociado a una línea horizontal.

Aunque las técnicas de detección de puntos y líneas pueden ser bastante útiles en muchos casos, la estrategia más común para encontrar discontinuidades en imágenes de niveles de grises es la detección de bordes. Esto se debe a que, en la mayoría de las aplicaciones, no se trata con imágenes que incluyan puntos o líneas claramente definidas.

2.3. Técnicas basadas en detección de bordes

Un *borde* se define como la frontera entre dos regiones con niveles de grises relativamente diferentes. La detección de bordes reduce significativamente la cantidad de información irrelevante en una imagen, conservando, al mismo tiempo, las características fundamentales de su estructura. Existen muchas técnicas diferentes para la detección de bordes pero, en la mayoría de los casos, la idea básica consiste en aplicar localmente algún operador derivativo como el gradiente o el Laplaciano. En la figura 2.6 se ilustra este concepto. En la parte izquierda de la figura se puede ver una imagen con una banda clara sobre un fondo negro y debajo de ésta se observan el perfil de los niveles de grises en una línea horizontal, y la primera y segunda derivadas de ese perfil. Las subfiguras de la derecha representan las mismas magnitudes para una imagen con una banda oscura sobre fondo claro. Obsérvese que el perfil de los bordes presenta cierta pendiente. Esto se debe a que, en general, los bordes en imágenes digitales están ligeramente difuminados a causa del pixelado.

En la figura 2.6(c) se observa que la primera derivada de la imagen de la izquierda presenta un máximo en la transición de oscuro a claro y un mínimo en la transición opuesta. Como es lógico, es cero en las zonas del perfil que tienen un nivel de gris constante. La segunda derivada (figura 2.6(d)) es positiva en la parte de la transición asociada al lado oscuro del borde, negativa para la parte asociada al lado claro del borde y cero en las áreas en las que el nivel de gris es constante. Por lo tanto, es posible usar la magnitud de la primera derivada para detectar la presencia de un borde en una imagen, y el signo de la segunda derivada para determinar si un píxel cercano a ese borde se encuentra en la parte clara u oscura del borde. El hecho de que la segunda derivada tenga pasos por cero en el punto intermedio de las transiciones es importante. Como veremos más adelante, esta circunstancia es útil para fijar con exactitud la posición de los bordes presentes en una imagen.

De entre las múltiples estrategias para la detección de bordes que podemos encontrar, en esta sección repasaremos técnicas basadas en operadores derivativos como el gradiente y el Laplaciano, el método para el enlazado de bordes mediante la transformada Hough y, por último, el famoso algoritmo de segmentación de Canny [11].

2.3.1. Detección de bordes por gradiente. El operador Sobel

El gradiente de una imagen $f(x, y)$ en un punto (x, y) es el vector

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} . \tag{2.5}$$

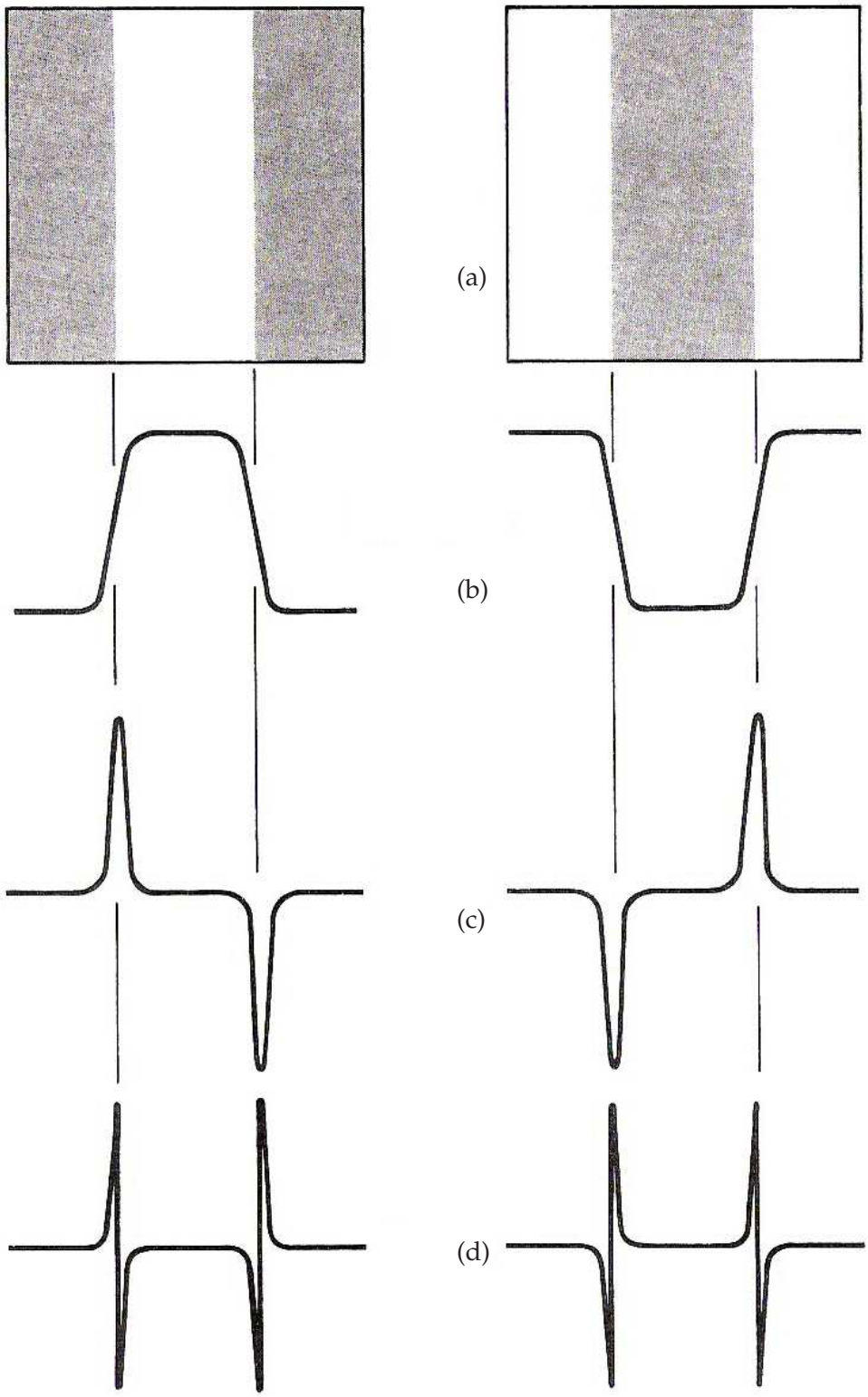


Figura 2.6.: Detección de bordes mediante operadores derivativos. (a) Imagen. (b) Perfil de una línea horizontal. (c) Primera derivada. (d) Segunda derivada. Nótese que la segunda derivada tiene pasos por cero justo en la posición del borde.

Como sabemos, el vector gradiente apunta en la dirección de máxima variación de f en el punto (x, y). La magnitud de este vector será

$$|\nabla f| = [G_x^2 + G_y^2]^{1/2} \ , \tag{2.6}$$

que indica la máxima variación por unidad de distancia de $f(x, y)$ en la dirección de ∇f. No obstante, lo más habitual es aproximar el gradiente mediante valores absolutos:

$$|\nabla f| \approx |G_x| + |G_y| \ , \tag{2.7}$$

que, además, es mucho más fácil de implementar, especialmente cuando estamos tratando con imágenes digitales.

La dirección del gradiente es también un dato importante. Llamando $\alpha(x, y)$ al ángulo de dirección del gradiente del vector ∇f en el punto (x, y). Se puede calcular mediante:

$$\alpha(x, y) = \arctan\left(\frac{G_x}{G_y}\right) \ , \tag{2.8}$$

donde el ángulo α se mide respecto al eje x.

Nótese que el cálculo del gradiente de una imagen se basa en la obtención de las derivadas parciales $\partial f/\partial y$ en cada uno de los píxeles que la componen. Existen muchas formas de implementar operadores derivativos digitales, sin embargo, los *operadores Sobel* destacan gracias a su efecto de suavizado. Esta característica es importante ya que, típicamente, los operadores derivativos tienden a incrementar el ruido.

La implementación del operador Sobel también se basa en el uso de máscaras. Éstas se aplican sobre la imagen mediante una convolución espacial de forma que se va recorriendo la imagen píxel a píxel hasta que se genera una imagen de salida completa. Generalmente se usan dos máscaras de tamaño 3×3 como las de la figura 2.7, cada una de ellas detecta los bordes en dirección ortogonal a su complementaria. A estas máscaras se las conoce como *operadores Sobel*. La máscara de la parte superior representa una zona 3×3 de la imagen que estamos tratando, las z representan los niveles de gris de los píxeles de la imagen.

Dadas las máscaras de la figura 2.7, las componentes del gradiente G_x y G_y en el píxel z_5 vendrán dadas por las ecuaciones 2.9:

$$\begin{aligned} G_x &= (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \ , \\ G_y &= (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \ . \end{aligned} \tag{2.9}$$

Una vez obtenidas las componentes G_x y G_y se puede calcular el valor del gradiente en el píxel que estemos evaluando usando la ecuación 2.7. Para obtener el gradiente del siguiente píxel

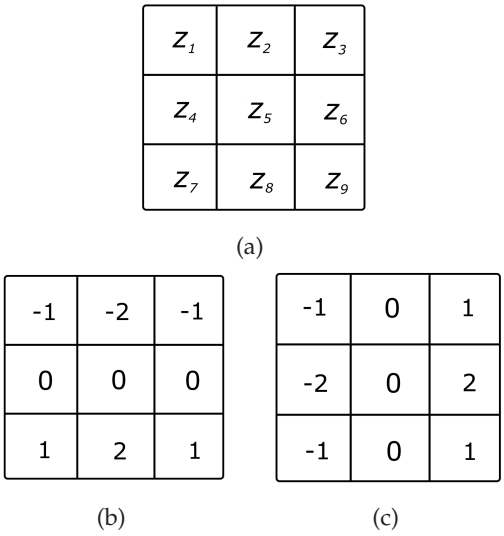


Figura 2.7.: (a) Región 3×3 de la imagen. (b) Máscara usada para calcular G_x en el punto central de (a). (c) Máscara usada para calcular G_y en el mismo punto. A este par de máscaras se las conoce como operadores Sobel

las máscaras han de desplazarse hasta éste y repetir todo el proceso. Cuando se han recorrido todos los píxeles que componen la imagen, la salida es una imagen gradiente del mismo tamaño que la imagen original.

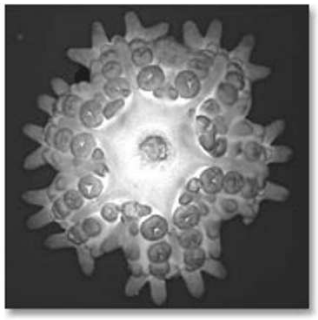
En la figura 2.8 podemos ver los resultados obtenidos tras aplicar los operadores Sobel sobre una imagen de niveles de grises. En este caso sólo hemos aplicado las máscaras para detectar bordes horizontales y verticales, sin embargo, también pueden definirse máscaras para detectar bordes oblicuos.

Los resultados obtenidos tras aplicar las máscaras adicionales para la detección de bordes oblicuos pueden apreciarse en la figura 2.9. Nótese que, para algunas imágenes, como la de este ejemplo, puede resultar muy beneficioso el uso de estas máscaras. Esto se debe a que la imagen presenta bordes en todas las direcciones a causa de su silueta circular, sin embargo, en aquellas aplicaciones en las que los objetivos tienen formas rectangulares su implementación no es en absoluto necesaria.

2.3.2. Detección de bordes mediante Laplaciano

El Laplaciano de una función bidimensional $f(x, y)$ es una derivada de segundo orden que se define como:

$$\nabla^2 f = \frac{\partial^2 f}{\partial^2 x^2} + \frac{\partial^2 f}{\partial^2 y^2} \, . \tag{2.10}$$



(a)

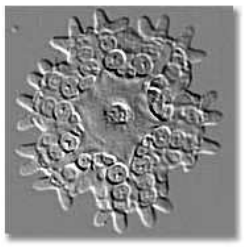
(b)

(c)

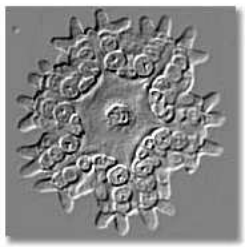
Figura 2.8.: (a) Imagen original. (b) Bordes horizontales. (c) Bordes verticales.

(a)

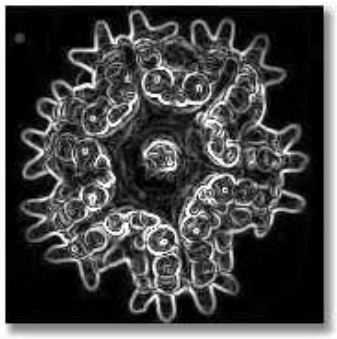
(b)



(c)



(d)



(e)

Figura 2.9.: Bordes obtenidos tras aplicar la máscara de dirección: (a) 45º; (b) 135º; (c) 225º; (d) 315º; (e) Resultado final, suma de todas las componentes

Como ocurre en el caso del gradiente, la ecuación 2.10 puede implementarse de diversas formas. Para el caso de una región de 3×3 píxeles, la forma más frecuente en la práctica es

$$\nabla^2 f = 4z_5 - (z_2 + z_4 + z_6 + z_8) \, , \tag{2.11}$$

donde el significado de los coeficientes z ya ha sido definido en la sección anterior. La condición principal al definir un Laplaciano digitalmente es que el coeficiente asociado al píxel central debe ser positivo y los coeficientes asociados a los píxeles vecinos debe ser cero (ver figura 2.10). Por ser un operador derivativo, la suma de todos los coeficientes de un Laplaciano debe ser cero. Por lo tanto, la salida será cero siempre que el píxel evaluado y sus vecinos tengan el mismo valor. En la figura 2.10 se muestra una máscara que puede servir para implementar la ecuación 2.11.

0	-1	0
-1	4	-1
0	-1	0

Figura 2.10.: Máscara para implementar el Laplaciano

Aunque, como hemos visto antes, el Laplaciano responde ante transiciones en la intensidad casi nunca se usa como detector de bordes por varias razones. En primer lugar, el Laplaciano presenta una inaceptable sensibilidad al ruido por ser una derivada de segundo orden. Además, el Laplaciano produce bordes dobles (ver figura 2.6) y no es capaz de detectar la dirección de la transición. Por estas razones, el Laplaciano es más comúnmente usado para decidir si un píxel se encuentra en la zona oscura o clara de un borde previamente detectado.

Una forma más común de usar el Laplaciano para la localización de bordes es usar sus cruces por cero (ver figura 2.6). El primer operador basado en los pasos por cero fue inicialmente propuesto por Marr y Hildreth [9]. Estos sugirieron que, para detectar los cambios de intensidad (bordes) de forma efectiva, el operador debía poseer dos cualidades: En primer lugar, debía ser un operador derivativo, ya fuera de primer o segundo orden. Y en segundo lugar, dicho operador debía ser ajustable a cualquier escala de forma que mediante filtros más grandes se pudieran detectar los bordes más difuminados y usando filtros más pequeños se pudieran detectar los bordes más abruptos. Esto llevó a lo que se conoce como el operador *Laplaciano de Gaussiana* o *LoG*. Este operador usa las propiedades de suavizado de la función Gaussiana al mismo tiempo que efectúa una derivada de segundo orden mediante un Laplaciano digital. Finalmente, la localización de los bordes se hace identificando los cruces por cero de la imagen de salida.

Esta técnica se efectúa convolucionando la imagen con la Laplaciana de una función Gaus-

siana de dos dimensiones. La función Gaussiana es de la forma

$$h(x,y) = exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) , \tag{2.12}$$

donde σ es la desviación estándar. Haciendo el cambio de variable $r^2 = x^2 + y^2$ y aplicando la ecuación 2.10 obtenemos el Laplaciano de h, es decir, la segunda derivada de h respecto de r:

$$\nabla^2 h = \left(\frac{r^2-\sigma^2}{\sigma^4}\right) exp\left(-\frac{r^2}{2\sigma^2}\right) . \tag{2.13}$$

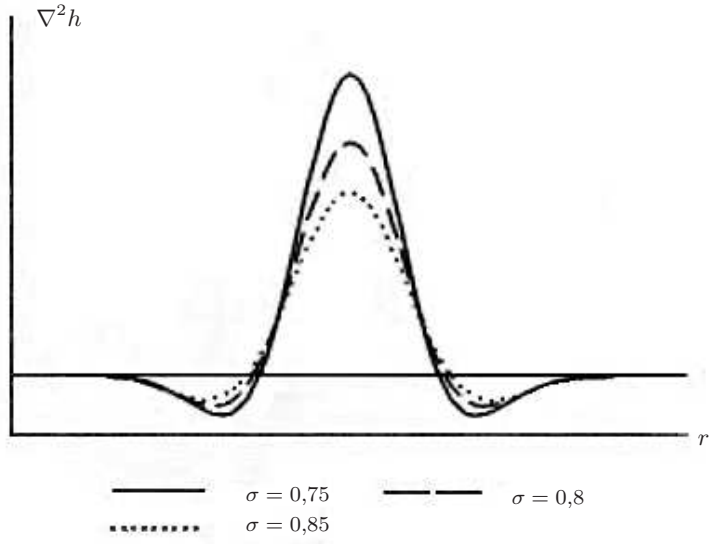
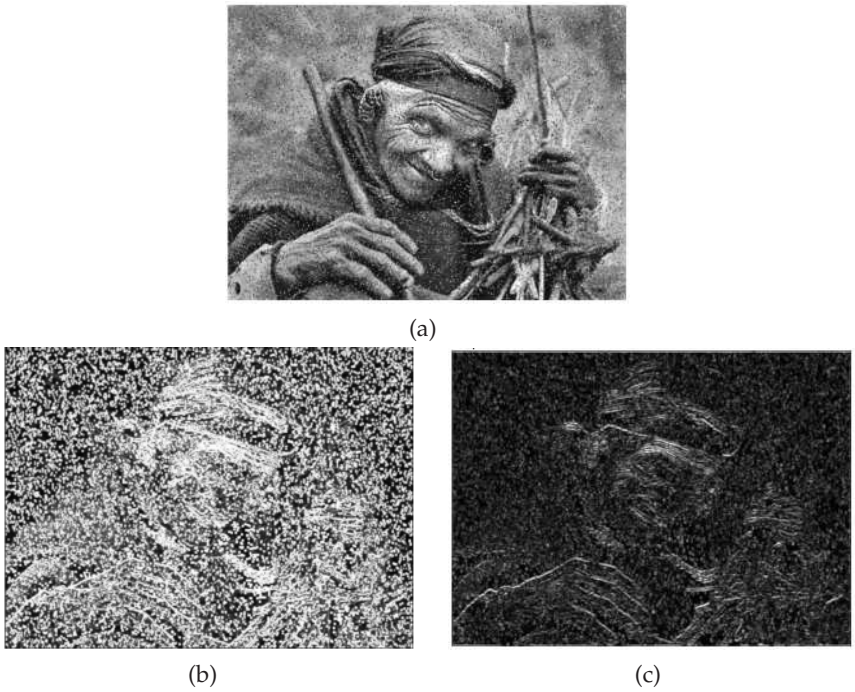


Figura 2.11.: Sección transversal de $\nabla^2 h$. El valor de σ es proporcional al suavizado de la imagen

La figura 2.11 muestra la sección transversal de la función $\nabla^2 h$. Esta función, vista en 3D, presenta simetría circular y tiene forma de típico sombrero mexicano. El eje vertical se corresponde con la intensidad. Simplemente observando la forma de esta función podemos intuir su efecto al aplicarla sobre una imagen. El uso de la función Gaussiana provoca un suavizado en la imagen que reduce el efecto del ruido (ver figura 2.12). Por otra parte, ensancha los bordes, es decir, hace las transiciones entre diferentes intensidades mucho menos abruptas. Esto puede parecer perjudicial ya que precisamente lo que queremos es encontrar esas transiciones. Sin embargo, al ser el Laplaciano un operador derivativo de segundo orden, encuentra el lugar exacto dentro de ese borde difuminado y ese lugar es, precisamente, el paso por cero (ver figura 2.6).

El efecto de suavizado se puede ajustar mediante el parámetro σ, que controla la anchura de la función. Valores altos de σ generan funciones más anchas y, por lo tanto, producen un suavizado mayor. En la figura 2.13 se puede ver un ejemplo de detección de bordes para dos valores diferentes de σ. Obsérvese cómo el valor de este parámetro puede usarse para controlar el nivel de detalle de los bordes detectados. De esta forma, además de reducir el efecto del ruido, puede ajustarse la sensibilidad del operador *LoG* en función de la imagen de entrada. Nótese que a la



(a)

(b) (c)

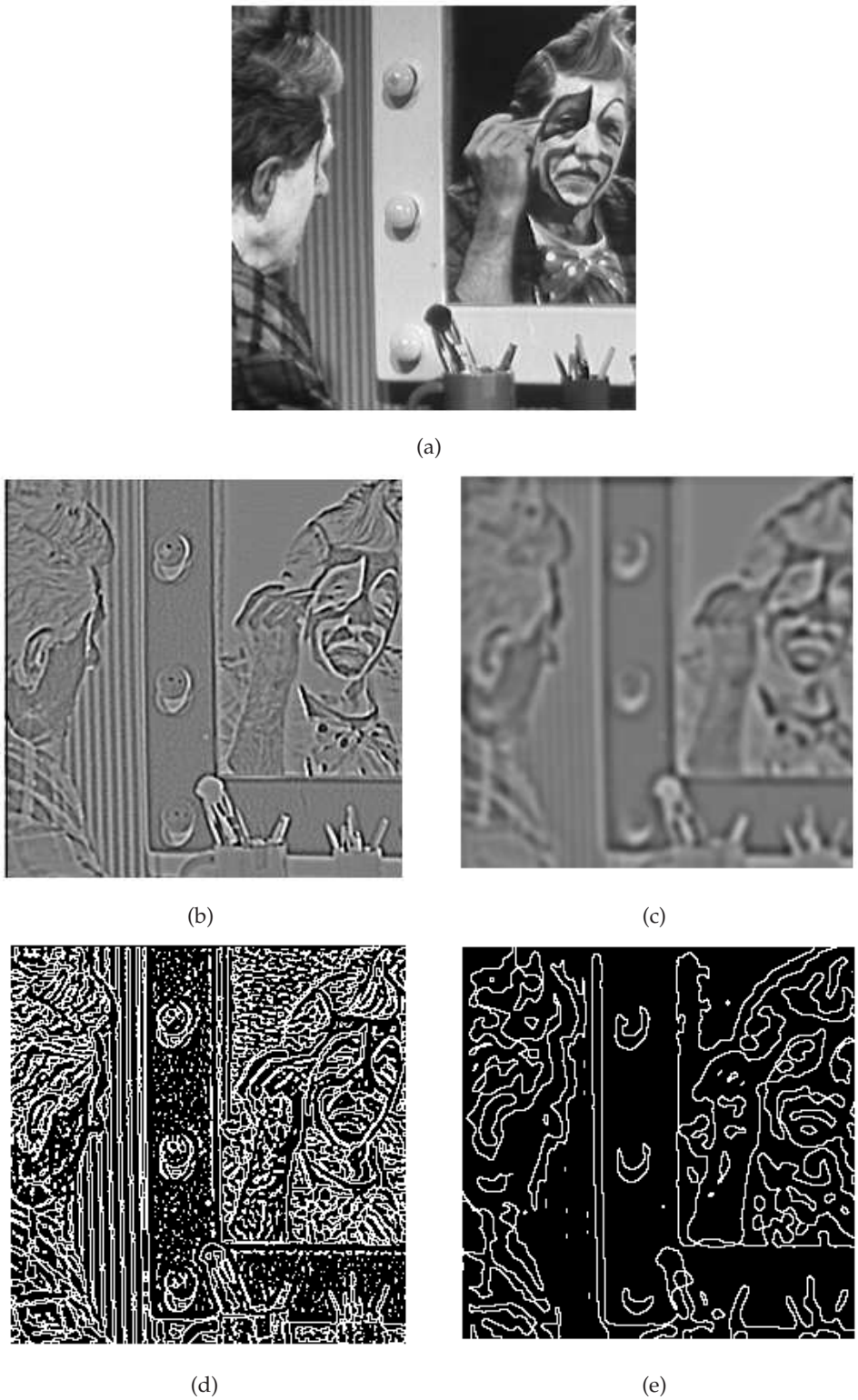
Figura 2.12.: (a) Imagen ruidosa. (b) Detección de bordes mediante Laplaciano. El efecto del ruido es inaceptable. (c) Detección de bordes mediante LoG. Aunque el resultado no es del todo satisfactorio el efecto de suavizado mejora los bordes obtenidos.

salida del operador, es necesario un sencillo postprocesado de la imagen para encontrar los cruces por cero en la imagen de salida, cuyos píxeles tienen valores positivos y negativos.

Las máscaras usadas para implementar digitalmente el operador *LoG* son muy similares a la de la figura 2.10, con la diferencia de que suelen ser más grandes para poder aproximar mejor el modelo de la figura 2.11 y controlar el parámetro de suavizado.

De todo lo anterior se puede concluir que la técnica de detección de bordes mediante el Laplaciano combinado con la Gaussiana es una buena alternativa en imágenes con bordes borrosos o un nivel alto de ruido. Gracias a los pasos por cero, se pueden localizar los bordes con bastante precisión y las propiedades de suavizado de $\nabla^2 h$ reducen considerablemente el efecto del ruido. La principal desventaja de este método es que se incrementan considerablemente el coste computacional y el tiempo de proceso.

Una solución alternativa de menor coste computacional es la técnica llamada *Diferencia de Gaussianas* o *DoG*. Esta estrategia consiste en aproximar la función $\nabla^2 h$ mediante dos gaussianas de diferente σ. Si se aplican estas dos gaussianas sobre la misma imagen obtendremos dos imágenes con diferente grado de difuminado. Restando estas dos imágenes entre sí se consigue una salida similar a las mostradas en la figura 2.13 con un poco menos de tiempo de proceso. Además,



(a)

(b)

(c)

(d)

(e)

Figura 2.13.: Efecto del valor de σ en la detección de bordes. (a) Imagen original. (b) Salida normalizada del operador LoG con $\sigma = 1{,}0$. La verdadera salida tiene valores negativos y positivos. (c) Salida normalizada del operador LoG con $\sigma = 3{,}0$. Nótese el difuminado. (d) Bordes detectados tras encontrar los pasos por cero en (b). (e) Bordes tras encontrar los pasos por cero en (c).

Marr [10] demostró que la visión humana se basa en un concepto muy similar al DoG y la técnica de encontrar los "pasos por cero".

2.3.3. Enlazado de bordes. La transformada Hough

Las técnicas de detección de bordes que hemos analizado hasta ahora tratan de identificar los píxeles que se encuentran entre zonas de características diferentes. Se espera siempre que el resultado sea un borde continuo que separe totalmente unas regiones de otras pero, desafortunadamente, esto no ocurre casi nunca. A causa del ruido (siempre presente en mayor o menor medida) o quizás por la naturaleza de la imagen y el "afinado" del algoritmo detector, siempre quedan bordes inconexos entre sí. En la mayoría de las aplicaciones esto no es deseable y se hace necesario el postprocesado de la segmentación para conseguir bordes que definan unos contornos cerrados o identifiquen alguna forma geométrica que podría ser el objetivo final de la segmentación. Básicamente, existen dos formas diferentes de abordar este problema: mediante procesado local de la imagen o mediante procesado global (transformada Hough).

La forma más simple de enlazar bordes es mediante el *procesado local* de la imagen. Esto es, analizando las características de los píxeles contenidos en una región relativamente pequeña respecto al tamaño total de la imagen (por ejemplo,regiones de 3×3 ó 5×5 píxeles). Para cada píxel (x, y) que haya sido sometido a una detección de bordes, se analizan las similitudes con sus vecinos contenidos en esa región de acuerdo con dos criterios:

1. La magnitud del operador gradiente usado para producir los bordes en ese punto (ver ecuación 2.6).
2. La dirección del vector gradiente en ese punto.

Por lo tanto, diremos que un píxel (x', y') previamente definido como "borde" es similar a otro píxel (x, y) cercano a él cuando

$$|\nabla f(x, y) - \nabla f(x', y')| \leq T \ , \tag{2.14}$$

donde T es un umbral no negativo.

La dirección del vector gradiente viene determinada por la ecuación 2.8. Luego diremos que un píxel (x', y') previamente marcado como "borde" es similar a otro píxel (x, y) cercano a él si

$$|\alpha(x, y) - \alpha(x', y')| < A \ , \tag{2.15}$$

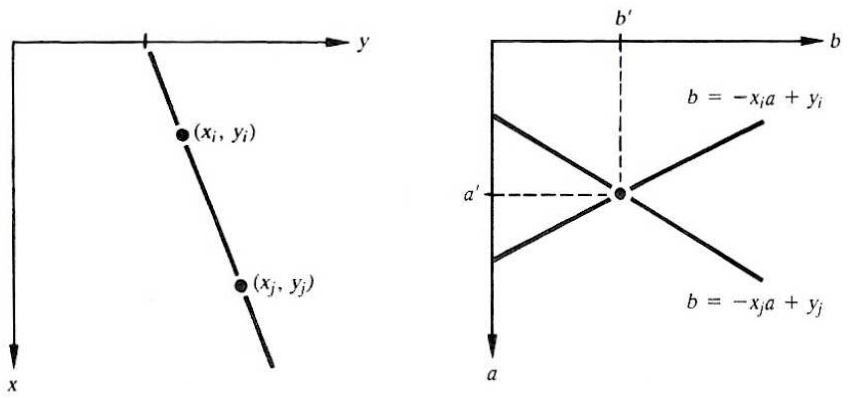
donde A es un ángulo umbral.

Si el píxel (x, y) satisface los dos criterios de similitud será marcado como píxel "borde" y enlazado al píxel (x', y').

Este método de procesado local puede ser útil en muchos casos en que los bordes son rectangulares o mantienen unas direcciones fáciles de predecir. Sin embargo, al ser un método basado en procesado local, necesita que los bordes a conectar estén suficientemente próximos entre sí. Los métodos basados en procesado global ofrecen una alternativa más sofisticada y versátil. Uno de los métodos más extendidos para el enlazado de bordes mediante *procesado global* de la imagen es el de la *transformada Hough*.

Esta técnica para enlazado de bordes consiste en determinar si una serie de puntos se ajustan a una curva de una determinada forma. Para simplificar la siguiente explicación consideramos el caso particular de que esa curva es, en realidad, una recta, aunque modificando las ecuaciones usadas se pueden modelar otras formas.

Consideremos un punto (x_i, y_i) y una recta definida mediante ese punto y su pendiente de la forma $y_i = ax_i + b$. Existen infinitas rectas que pasan por el punto (x_i, y_i) y satisfacen la ecuación $y_i = ax_i + b$ para infinitos valores de a y b. Sin embargo, si escribimos esta ecuación de la forma $b = -x_i a + y_i$ y definimos el plano ab (también llamado *espacio de parámetros*) tenemos la ecuación de una *única* recta para un punto (x_i, y_i) fijo. Además, para un segundo punto (x_j, y_j) también existe una recta en el espacio de parámetros ab asociada con él. Esta segunda recta se cruza con la recta asociada a (x_i, y_i) en (a', b'), donde a' representa la pendiente y b' el desplazamiento de la recta que contiene a los puntos (x_i, y_i) y (x_j, y_j) en el plano xy. De hecho, todos los puntos contenidos en la recta del espacio xy tienen rectas asociadas a ellos en el espacio de parámetros ab. Por lo tanto, cada punto en el espacio xy de la imagen se corresponde a una recta en el espacio de parámetros ab y cada punto en el espacio ab se corresponde con una recta en el espacio xy. La figura 2.14 ilustra este concepto.



(a) (b)

Figura 2.14.: (a)Plano xy. (b) Espacio de parámetros ab.

Una vez establecida esta correspondencia entre los dos dominios es fácil comprender como funciona el método de la transformada Hough. El espacio de parámetros se divide en celdas llamadas *acumuladores* y cumplen la función de contadores (ver figura 2.15). Inicialmente todas las celdas contador están a cero. Por cada punto (x_k, y_k) de la imagen que se evalúa (puntos previamente marcados como bordes por algún otro método de detección) se genera una recta en el espacio de parámetros. Como se observa en la figura 2.14, cada punto de intersección entre rectas del espacio de parámetros representa una línea en la imagen. Cuando uno de estos puntos de intersección es común a muchas rectas en el espacio ab significa que existen muchos puntos alineados en el plano xy (esto es, en la imagen). El contador de la celda aumentará proporcionalmente sugiriendo, por lo tanto, que en la imagen existe un borde con forma de línea.

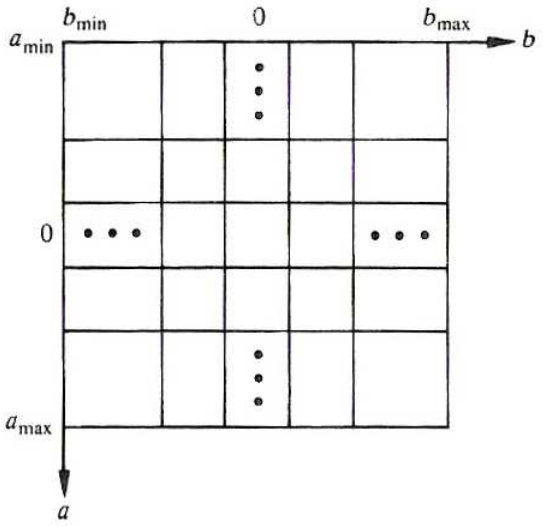


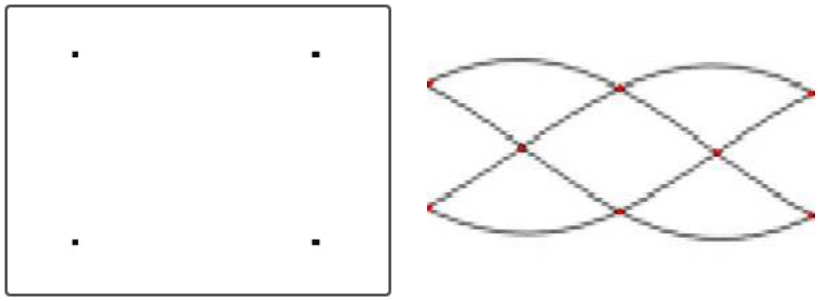
Figura 2.15.: División del espacio de parámetros en celdas acumuladoras. (a_{max}, a_{min}) y (b_{max}, b_{min}) indican el rango de valores esperado para las pendientes y el desplazamiento de las rectas.

La precisión en la colinearidad de los puntos evaluados la determina el número de subdivisiones del espacio de parámetros. Esta es una de las características más ventajosas de esta técnica ya que permite controlar el coste computacional del algoritmo, que aumenta de forma *lineal* con el número de celdas. A mayor número de celdas, más tiempo de proceso y también mayor resolución. Hay que tener en cuenta que no siempre será deseable una resolución máxima ya que en muchos casos puede ser conveniente agrupar varios puntos en la misma línea aunque éstos no estén estrictamente alineados.

Hasta ahora hemos usado la forma $y = ax + b$ para representar las rectas pero en la práctica esta formulación presenta un problema. Para líneas verticales, el parámetro de la pendiente tiende a infinito. Una forma de soslayar este inconveniente es usar una representación diferente para las rectas:

$$x \cos\theta + y \sin\theta = \rho \; . \tag{2.16}$$

La subdivisión del espacio de parámetros $\rho\theta$ se mantiene igual que con la ecuación punto-pendiente. La diferencia es que, ahora, al trasladar los puntos del plano xy al espacio de parámetros aparecen curvas sinusoidales en lugar de rectas (ver figura 2.16). Al igual que antes, si tenemos M puntos alineados en el espacio xy, en el espacio de parámetros aparecerán M curvas sinusoidales que se intersectan en (ρ_i, θ_i). Encontrar los acumuladores con valores más altos nos permitirá encontrar los valores de ρ y θ que definen las líneas en la imagen.



(a) (b)

Figura 2.16.: (a) Imagen en el plano xy. (b) Curvas de la transformada Hough mostrando los seis puntos correspondientes a las seis rectas que pueden pasar por los cuatro puntos de (a).

En la figura 2.16 se ilustra con un ejemplo como funciona la transformada de Hough. La imagen está formada por cuatro puntos correspondientes a las esquinas de un cuadrado. Estos cuatro puntos dan lugar a cuatro sinusoides en el espacio $\rho\theta$. Las cuatro sinusoides se cortan en seis puntos (en la figura 2.16(b) aparecen ocho puntos, pero hay que recordar que los dos puntos para $\theta = 90^o$ son los mismos que los puntos para $\theta = -90^o$, por lo que, de ocho, son seis distintos), correspondientes a las seis rectas posibles que pasan por los cuatro puntos del plano xy que son, a saber, los cuatro lados del cuadrado y las dos diagonales.

(a) (b) (c)

Figura 2.17.: Unión de bordes mediante transformada Hough. (a) Imagen original. (b) Detección de bordes mediante el algoritmo de Canny. Esta es la entrada para la transformada de Hough. Nótese que, al estar los rectángulos superpuestos, el detector de Canny no puede completar las siluetas. (c) Resultado tras aplicar la transformada y representar en el dominio cartesiano

El uso de la transformada Hough puede extenderse a la detección de formas más complicadas. En realidad puede detectarse cualquier forma que pueda expresarse matemáticamente con un número finito de parámetros. Sin embargo, el número de parámetros determinará las dimensiones del acumulador aumentando así la complejidad del problema.

Para ilustrar la utilidad de la técnica de enlazado de bordes mediante la transformada Hough vamos a ver un ejemplo práctico. Supongamos que estamos diseñando una aplicación de visión artificial cuyo objetivo es extraer las siluetas rectangulares presentes en una imagen. Nuestra imagen de entrada será la que se muestra en la figura 2.17(a), en la que hay dos rectángulos superpuestos. Encontrar los bordes presentes entre los dos rectángulos es una tarea prácticamente imposible para cualquier algoritmo detector de bordes debido a la iluminación de la imagen. Como vemos, ni siquiera el detector de Canny (lo veremos más adelante) consigue separar los dos rectángulos (figura 2.17(b)). En esta situación, y puesto que disponemos del conocimiento a priori de que todos los bordes presentes en la imagen son líneas, podemos usar la transformada Hough para unir los bordes encontrados por el detector Canny.

Puede observarse que muchos bordes tienen más de una línea asignada. Esto se debe a que en el espacio de parámetros existen varios máximos (lugares de intersección entre las curvas generadas) próximos entre sí. Nótese también el hecho de que los bordes de la imagen con menos puntos (aquellos de menor longitud) son los que generan líneas más imprecisas. Existen técnicas para controlar este efecto así como para limitar la longitud de las rectas generadas.

2.3.4. El detector de bordes de Canny

El detector de bordes de Canny es para muchos el mejor método que existe para detectar discontinuidades en una imagen. En su archiconocido artículo [11], John Canny enfocó el problema de detección de bordes desde una perspectiva de procesado digital de señales. Su objetivo era diseñar un algoritmo óptimo de acuerdo con un determinado criterio. Para ello, especificó una función objetivo a optimizar y la usó para diseñar el algoritmo. Dicha función objetivo fue diseñada de forma que se obtuviera

- Mínima probabilidad de múltiples respuestas ante un único borde.
- Mínima probabilidad de dejar bordes sin detectar.
- Mínima distancia desde el borde marcado hasta el verdadero borde.

Los dos primeros criterios de la anterior lista hacen referencia al problema de *detección*, es decir, ante un borde presente en la imagen, el operador debe detectar ese borde y no otros. El tercer

criterio alude al problema de *localización*, esto es, cómo de preciso es el operador encontrando la verdadera posición de los bordes. Hay que decir que existe un compromiso entre detección y localización: cuanto más preciso es el algoritmo detectando bordes, menos preciso es localizándolos y viceversa.

El operador Canny trabaja en diferentes etapas. En primer lugar, se suaviza la imagen convolucionándola con una función Gaussiana de dos dimensiones. Esto sirve para filtrar el ruido presente en la imagen a costa de perder un poco de detalle. El grado de suavizado se puede controlar mediante la anchura de la Gaussiana. Como ya hemos comentado anteriormente, esta función se aproxima en la práctica mediante una máscara que recorrerá toda la imagen píxel a píxel. El tamaño de dicha máscara determinará el grado de suavizado.

Tras la operación de suavizado, se aplica algún operador derivativo de primer orden sobre la imagen para detectar los bordes y determinar la *fuerza* y dirección de éstos. Esto suele hacerse realizando el gradiente de forma espacial mediante operadores del tipo Sobel o similares. Las máscaras para realizar esta operación son las de la figura 2.7 y las ecuaciones 2.7 y 2.8 sirven para calcular la fuerza (magnitud) del borde y su dirección respectivamente.

Una vez detectados los bordes mediante gradiente, se inicia el proceso conocido como *supresión de no-máximos*. Habiendo calculado la magnitud de los bordes y su dirección, se realiza un *seguimiento* de éstos, eliminando aquellos que no tienen suficiente "fuerza". Este proceso de supresión sigue un ciclo de histéresis que se controla mediante dos umbrales T_1 y T_2, donde $T_1 > T_2$. El seguimiento sólo puede comenzar en un píxel-borde con un valor superior a T_1. El seguimiento del borde continúa en las dos direcciones a partir de ese píxel hasta que llega a un punto con un valor inferior a T_2 en donde se decide que el borde termina. Este mecanismo de histéresis ayuda a evitar que imágenes ruidosas den como resultado bordes entrecortados.

Resumiendo un poco, el algoritmo básico del detector de bordes de Canny actúa de la siguiente forma:

1. Aplicar la máscara de suavizado sobre la imagen.
2. Aplicar algún operador derivativo (Sobel) para encontrar bordes y calcular su magnitud y dirección.
3. Realizar la supresión de no-máximos sobre los bordes encontrados.

De todo lo expuesto anteriormente, vemos que el funcionamiento del detector de Canny puede controlarse mediante tres parámetros, la anchura de la máscara de suavizado y los dos umbrales T_1 y T_2. Si se incrementa la anchura de la máscara el detector será menos sensible al ruido pero

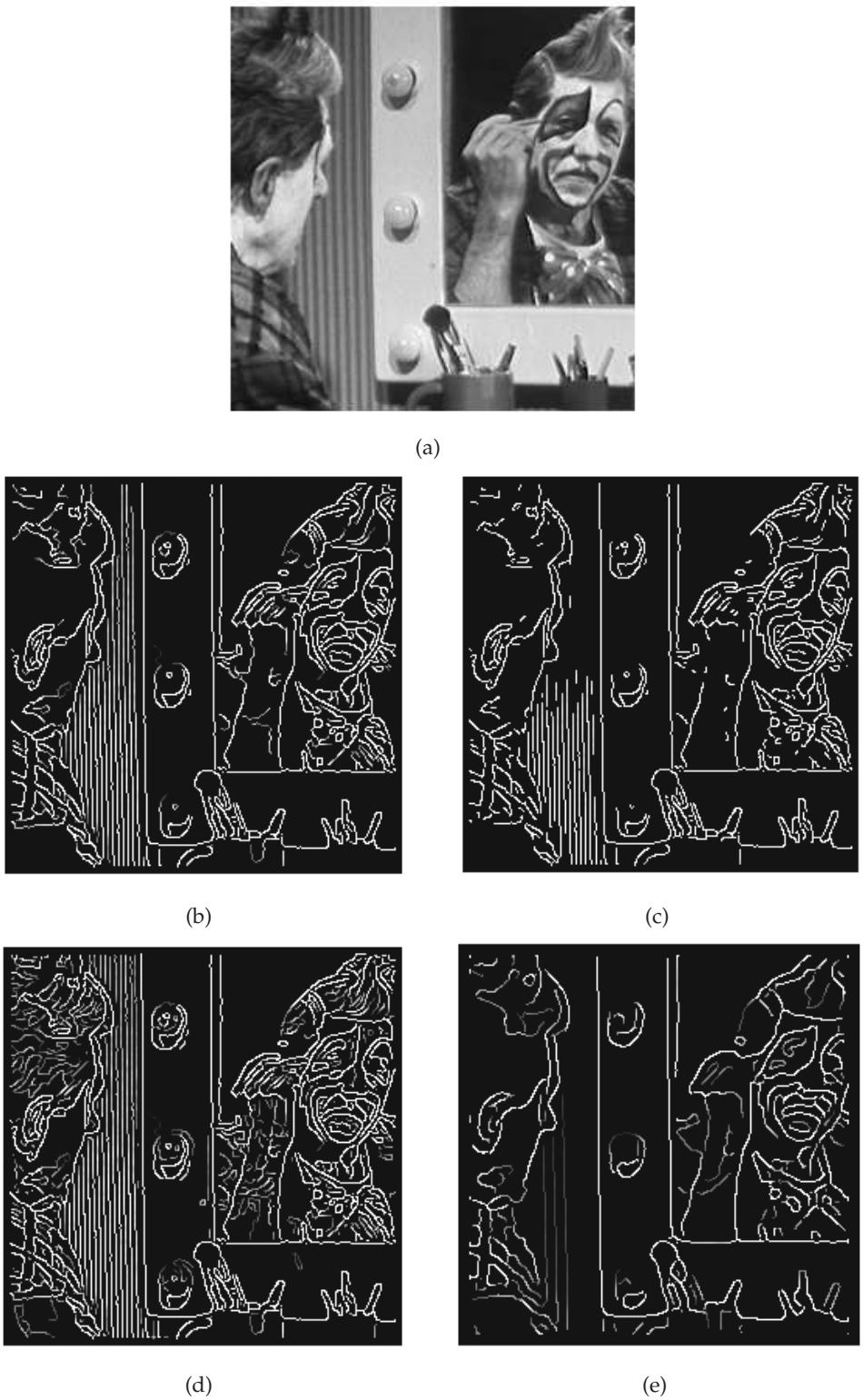
perderemos los detalles más finos de la imagen. Además, el uso de Gaussianas muy anchas también afecta a la localización exacta de los bordes. Normalmente, el umbral alto (T_1) suele dejarse bastante alto y el umbral bajo (T_2) bastante bajo para conseguir buenos resultados, aunque esto depende de la imagen de entrada y de los bordes objetivo en cada aplicación concreta. Generalmente, un umbral T_2 demasiado alto genera bordes fragmentados en imágenes con ruido y si el umbral T_1 se ajusta demasiado bajo aparecen demasiados bordes espurios.

En la figura 2.18 puede observarse la diferencia entre los resultados en función de los valores de los tres parámetros. La imagen de entrada es la misma que en la figura 2.13 sirvió para ilustrar el funcionamiento del operador LoG. La figura 2.18(b) ha sido obtenida usando una máscara Gaussiana con desviación estándar $\sigma = 1{,}0$ y umbrales $T_1 = 255$ y $T_2 = 1$. La mayoría de los bordes principales han sido detectados correctamente aunque este resultado es quizá demasiado detallado a causa del los valores de los umbrales. En la figura 2.18(c) el umbral inferior T_2 ha sido ajustado a 220, en este caso aparecen bordes fragmentados, lo que no suele ser deseable en la mayoría de las aplicaciones. La imagen 2.18(d) mantiene la $\sigma = 1{,}0$ y los umbrales son ahora $T_1 = 128$ y $T_2 = 1$, ahora aparecen los bordes más débiles. Nótese el detalle en el pelo del payaso. En la figura 2.18(e) los umbrales son iguales que en el caso anterior pero la Gaussiana tiene una desviación estándar $\sigma = 2{,}0$. El parámetro de suavizado tiene dos efectos: sirve para filtrar ruido y controlar el nivel de detalle que queremos captar. Se observa en este caso que los bordes detectados son menos ruidosos y se han perdido gran parte de los detalles. No obstante, los bordes principales se mantienen.

La figura 2.18 también nos sirve para ilustrar uno de los principales defectos de el algoritmo básico de Canny. A causa del seguimiento de bordes usando la dirección del gradiente y la supresión de no máximos, Canny es incapaz de detectar correctamente las uniones en forma de Y. El algoritmo interpreta estas uniones como un borde continuo y otro que se acerca sin llegar a unirlos. Este efecto se puede observar en la esquina inferior izquierda del espejo en la figura 2.18(b). Este problema se puede solucionar modelando este tipo de uniones en la implementación del seguimiento de bordes.

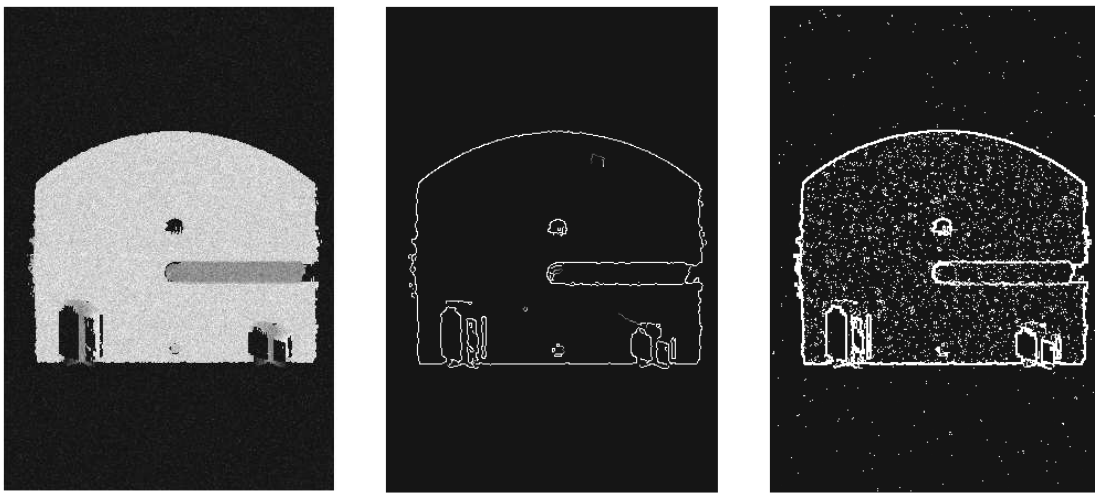
Para observar el comportamiento ante ruido del algoritmo de Canny frente a otros detectores más simples podemos echar un vistazo a la figura 2.19. Como vemos, mediante Canny se han detectado todos los bordes y se ha suprimido casi todo el ruido. El resultado usando el operador Sobel es mucho menos satisfactorio.

Por último, ilustraremos la capacidad del detector Canny para discriminar bordes mediante la figura 2.20. En la figura 2.20(a) tenemos una imagen de que queremos extraer únicamente los bordes correspondientes a su silueta. En una primera aproximación obtenemos el resultado que muestra la figura 2.20(b). Evidentemente, hay demasiados bordes detectados. El primer paso será aumentar el parámetro de suavizado. En la figura 2.20(c) se muestra el resultado. Han desapare-



(a) (b) (c) (d) (e)

Figura 2.18.: Detección de bordes mediante el operador Canny para diferentes valores de los parámetros. (a) Imagen original. (b) $\sigma = 1{,}0$, $T_1 = 255$, $T_2 = 1$. (c) $\sigma = 1{,}0$, $T_1 = 255$, $T_2 = 220$. (d) $\sigma = 1{,}0$, $T_1 = 128$, $T_2 = 1$. (e) $\sigma = 2{,}0$, $T_1 = 128$, $T_2 = 1$.



(a) (b) (c)

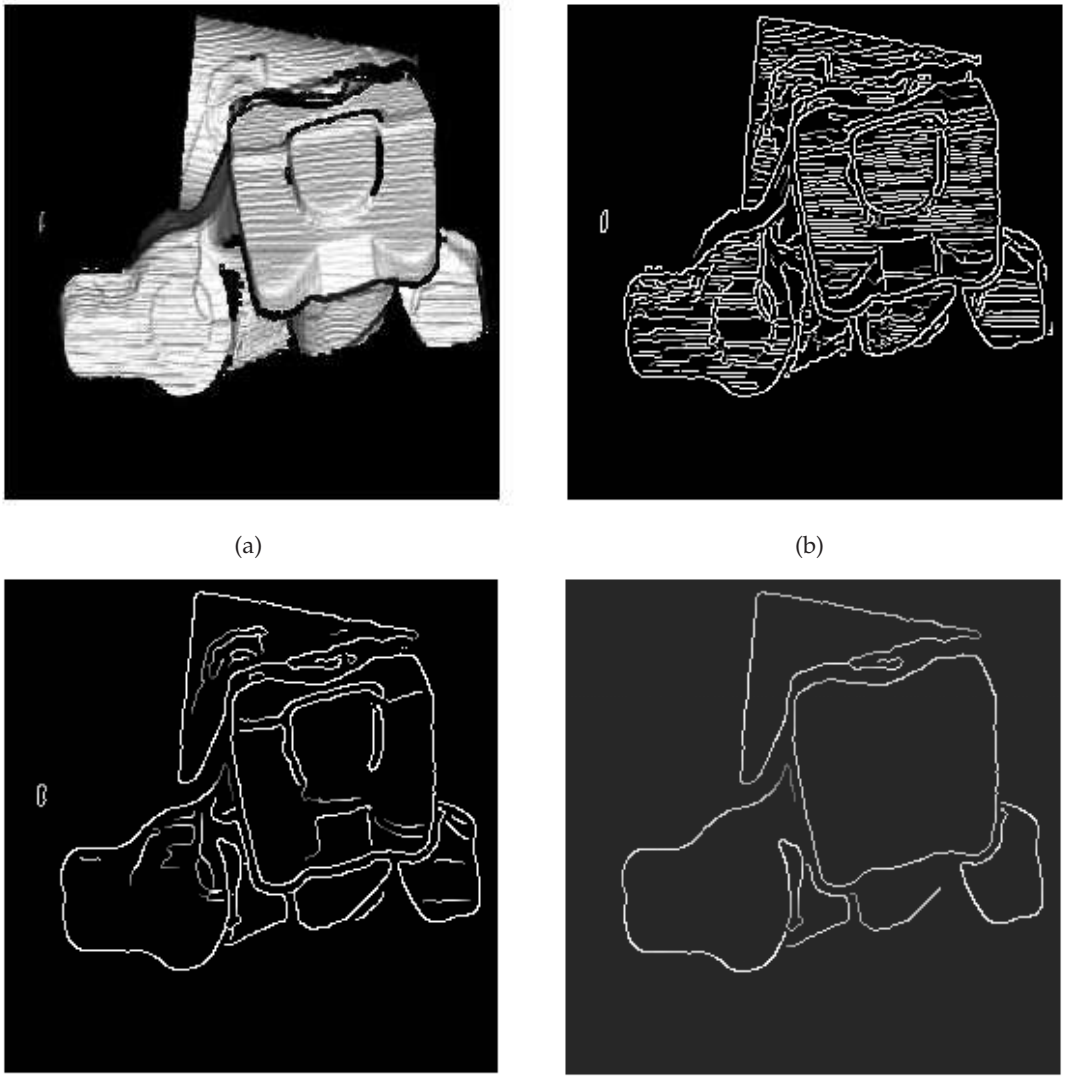
Figura 2.19.: Respuesta del operador Canny frente al ruido. (a) Imagen de entrada. (b) Detección de bordes con Canny ($\sigma = 1{,}0$). (c) Detección de bordes con Sobel y aplicando a la salida un umbral.

cido la mayoría de los bordes no deseados pero todavía quedan algunos debidos a las aristas en la pieza de la imagen. Aunque en la imagen original estos bordes son más débiles que los que corresponden a la silueta, en la imagen de salida tienen la misma fuerza debido a la saturación de la imagen. Esto se puede solucionar aplicando un escalado a los niveles de grises de la imagen original antes de aplicar el detector. Esto es lo que se ha hecho para obtener el resultado de la figura 2.20(d), en la que, además, se ha reducido el valor del umbral superior para suprimir los bordes más débiles.

2.4. Técnicas basadas en umbrales

Las técnicas basadas en umbrales son una de las aproximaciones más importantes para la segmentación de imágenes. En aquellas aplicaciones en las que el objetivo es distinguir un objeto del resto de la imagen (fondo) la técnica más rápida y sencilla es aplicar un umbral sobre el histograma de la imagen. Además, la mayoría de las técnicas de segmentación hacen uso de algún umbral cuyo valor, en último término, suele determinar el resultado final obtenido. Generalmente la imagen de entrada es de niveles de grises pero también puede aplicarse esta técnica sobre imágenes en color definiendo umbrales para cada canal *RGB*. En esta sección veremos varios ejemplos de umbralización de imágenes así como diferentes técnicas para obtener umbrales óptimos en función de la imagen a segmentar.

Hemos mencionado antes que las técnicas basadas en umbrales consisten en aplicar un umbral sobre el histograma de una imagen. Para ilustrar este concepto supongamos que tenemos un



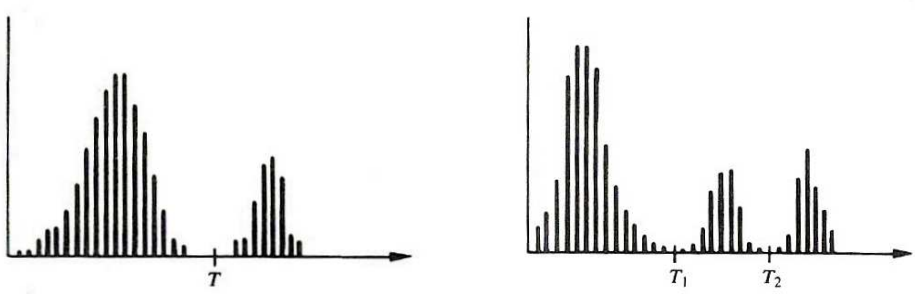
(a) (b)

(c) (d)

Figura 2.20.: Control de detalles con el operador Canny. (a) Imagen de entrada. (b) Salida con $\sigma = 1{,}0$, $T_1 = 255$, $T_2 = 1$. (c) Salida aumentando la desviación estándar ($\sigma = 1{,}8$). (d) Salida tras aplicar un escalado a la imagen original y bajar el umbral superior hasta $T_1 = 200$.

histograma de niveles de grises como el de la figura 2.21(a) que se corresponde con una imagen $f(x, y)$ compuesta por objetos claros sobre un fondo oscuro. Como se puede observar, el histograma presenta una distribución bimodal. La forma más obvia de separar los objetos del fondo es elegir un umbral T que separe los dos modos del histograma. Así, cualquier punto (x, y) que cumpla $f(x, y) > T$ será considerado un punto objeto y en cualquier otro caso será considerado fondo.

En la imagen de la figura 2.21(b) podemos ver un histograma que presenta tres modos dominantes. Este histograma podría corresponder a una imagen con dos objetos de diferentes niveles de grises sobre un fondo oscuro. En este caso podríamos clasificar un punto (x, y) de la imagen como perteneciente a un objeto cuando $T_1 < f(x, y) \leq T_2$. En el caso de $f(x, y) > T_2$ el punto sería considerado parte del otro objeto. El resto de los puntos $f(x, y) \leq T_1$ serían fondo. Este tipo de umbralización multinivel es, en general, menos fiable que el uso de un sólo umbral. Esto se debe a que establecer los umbrales adecuados para separar los diferentes objetos de la imagen suele ser complicado, especialmente cuando el número de modos del histograma es grande. Normalmente, para este tipo de problemas suele ser mejor opción el uso de un único umbral variable.



(a) (b)

Figura 2.21.: Histogramas de niveles de grises. (a) Partición del histograma mediante un único umbral. (b) Partición del histograma mediante dos umbrales.

Según lo expuesto anteriormente, puede decirse que el proceso de umbralización es una operación en la que se evalúa una imagen mediante una función T de la forma

$$T = T[x, y, p(x, y), f(x, y)] \, , \tag{2.17}$$

donde $f(x, y)$ es el nivel de gris en el punto (x, y) y $p(x, y)$ denota alguna propiedad local de este punto (por ejemplo, el nivel de gris de los vecinos de (x, y)). La imagen umbralizada $g(x, y)$ será

$$g(x, y) = \begin{cases} 1 & \text{si } f(x, y) > T \\ 0 & \text{si } f(x, y) \leq T \end{cases} \, . \tag{2.18}$$

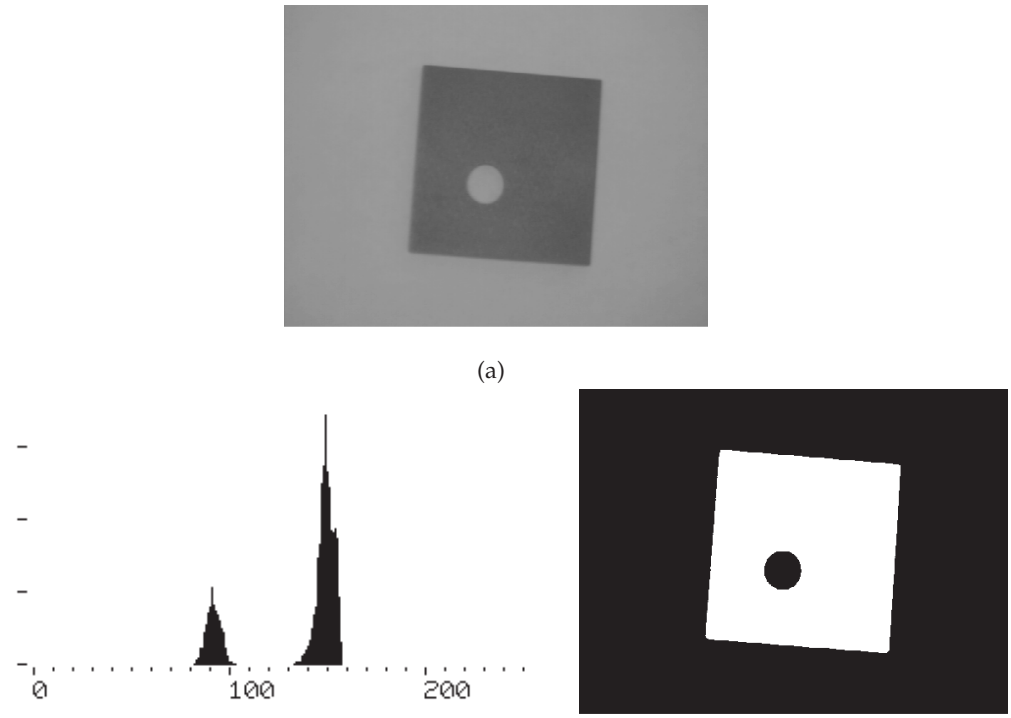
La imagen de salida es, por lo tanto, binaria. Los píxeles etiquetados con un 1 se corresponderán con objetos y los etiquetados con 0 se considerarán fondo.

Existen diferentes tipos de umbralización según el número de parámetros que usemos en la función T de la ecuación 2.17. Cuando T depende solamente de $f(x, y)$ el umbral es *global*. Si T depende de $f(x, y)$ y de $p(x, y)$ el umbral es *local*. Si además T depende de las coordenadas espaciales x e y, el umbral se considera *dinámico* o *adaptativo*.

2.4.1. Umbralización global

La forma más sencilla de segmentar una imagen mediante umbrales es usar un único umbral T tal y como muestra la figura 2.21(a). La segmentación se lleva a cabo recorriendo la imagen píxel a píxel y etiquetando cada uno como objeto o como fondo según el criterio establecido en la ecuación 2.18. Evidentemente, el éxito de esta estrategia de segmentación depende en gran medida de lo fácil que sea dividir el histograma mediante un único umbral.

La figura 2.22 muestra un ejemplo en el que el histograma de la imagen es fácilmente divisible en dos partes. La imagen de entrada es un objeto oscuro sobre un fondo más claro y el histograma correspondiente presenta, por lo tanto, una distribución bimodal. En este ejemplo basta con aplicar un sólo umbral para separar el objeto del fondo. La figura 2.22(c) muestra el resultado obtenido usando un umbral de $T = 120$.



(a)

(b)

(c)

Figura 2.22.: Segmentación de una imagen mediante umbralización global. (a) Imagen original. (b) Histograma de la imagen. (c) Imagen umbralizada con $T = 120$.

En la práctica, la segmentación de imágenes mediante esta técnica tan simple es sólo posible en entornos muy controlados donde las condiciones de iluminación permiten tratar con histogramas tan apropiados como los del ejemplo de la figura 2.22. Uno de estos entornos puede ser, por ejemplo, plantas industriales en donde se analizan objetos mediante visión artificial. En entornos de este tipo la iluminación puede ser controlada de forma que los histogramas sean lo más simples posible y la correlación entre éstos y los objetos de la imagen sea máxima. Para ver un ejemplo del efecto que produce la iluminación sobre los histogramas podemos echar un vistazo a la figura 2.23. Si observamos el histograma podemos ver que el gradiente en la iluminación de la imagen ha provocado que la frontera entre los dos modos bien separados que aparecían en el caso de la figura 2.22 se difumine. La segmentación de esta imagen mediante un único umbral global es imposible en este caso. Las figuras 2.23(c) y 2.23(d) muestran los resultados usando umbrales de $T = 80$ y $T = 120$ respectivamente. Veremos más adelante que pueden obtenerse resultados mucho más satisfactorios usando *umbrales adaptativos*.

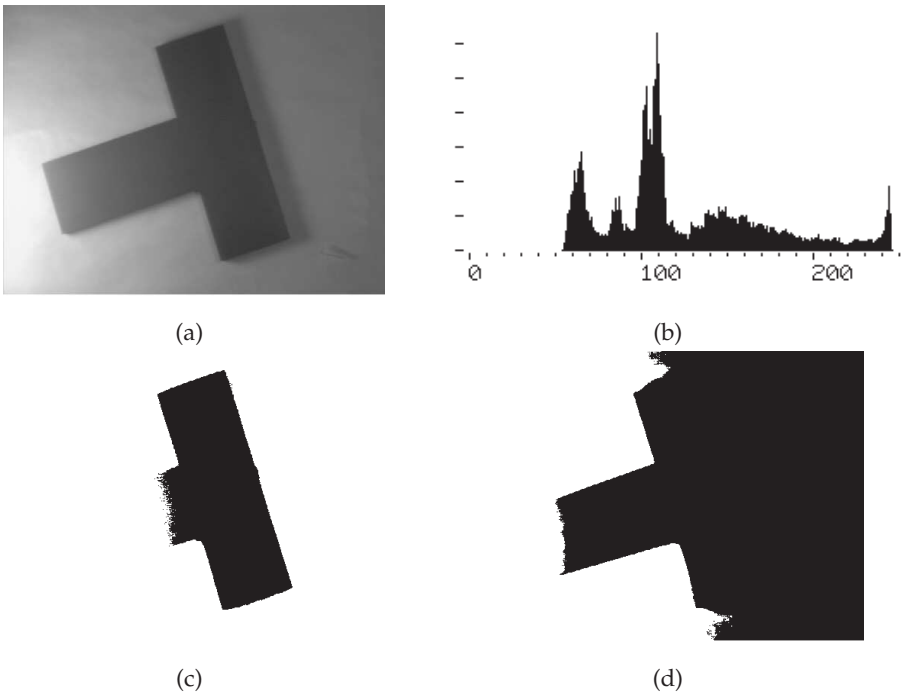
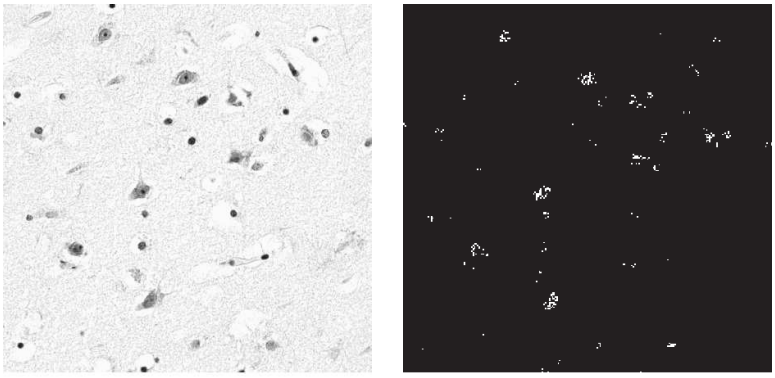


Figura 2.23.: Segmentación mediante umbralización global de una imagen con un severo gradiente en la iluminación. (a) Imagen de entrada. (b) Histograma de la imagen. (c) Segmentación con $T = 80$. (d) Segmentación con $T = 120$.

Una técnica que hace uso de los umbrales globales y es, además, muy popular en campos como la medicina y la geología es la conocida como *Density Slicing*. Esta técnica es también de muy sencilla aplicación. Se trata de usar dos umbrales en lugar de uno (ver figura 2.21(b)) de forma que separamos aquellos píxeles que se encuentran dentro de una banda de intensidades de grises (*grey density*). Usando varios pares de umbrales la imagen puede dividirse en varias capas (*slices*) de forma que los diferentes elementos de interés sean fácilmente identificables.

En la figura 2.24 vemos un ejemplo en el que esta técnica es útil. La figura 2.24(a) es una muestra de tejido cerebral en la que hay neuronas (manchas grises con el núcleo negro en el centro) y células de Glía (los círculos negros aislados) sobre el fondo claro. Si nuestro objetivo es detectar el porcentaje de neuronas presentes en la muestra lo que debemos hacer es encontrar y separar las manchas grises del resto de la imagen ignorando los círculos negros. La figura 2.24(b) muestra el resultado tras aplicar dos umbrales $T_1 = 130$ y $T_2 = 150$. Puede verse como las células de Glía no aparecen en la imagen de salida. Esta discriminación no hubiera sido posible mediante un sólo umbral.



(a) (b)

Figura 2.24.: Aplicación de Density Slicing sobre una muestra de tejido cerebral. Los objetos a detectar son las manchas grises. (a) Imagen original. (b) Resultado tras detectar los píxeles entre $T_1 = 130$ y $T_2 = 150$.

2.4.2. Umbralización adaptativa

Hemos visto en la sección anterior que el umbralizado global usa un sólo umbral para todos los píxeles de la imagen. La umbralización adaptativa, sin embargo, cambia el umbral dinámicamente a lo largo de la imagen. El umbralizado adaptativo exige calcular un umbral para cada píxel de la imagen. Si el valor de intensidad del píxel supera ese umbral será considerado como un objeto y si no, será considerado fondo. Esta técnica más sofisticada permite umbralizar imágenes con condiciones de iluminación problemáticas tales como gradientes o sombras.

Para determinar dicho umbral existen dos estrategias principales: la estrategia de *Chow y Kaneko* y el *umbralizado local*. Ambos métodos se basan en la hipótesis de que es más probable que la iluminación sea uniforme en pequeñas regiones de la imagen que en la imagen completa.

La estrategia de Chow y Kaneko [17] divide la imagen en regiones que se solapan entre sí. Para cada región se estima un umbral basado en el histograma de esa región. Dicha estimación implica la minimización de una función de error objetivo basada en ciertas suposiciones a priori

que se hacen sobre la imagen y en algunos casos no llega a converger a un valor satisfactorio. Al estar solapadas las regiones entre sí, se usan los valores de las regiones vecinas para interpolar el umbral para las regiones en las que no se ha conseguido resolver un umbral. Una vez conseguido un umbral para cada región se calcula un umbral para cada píxel de la imagen interpolando los valores de los umbrales de las regiones que se encuentran en su entorno.

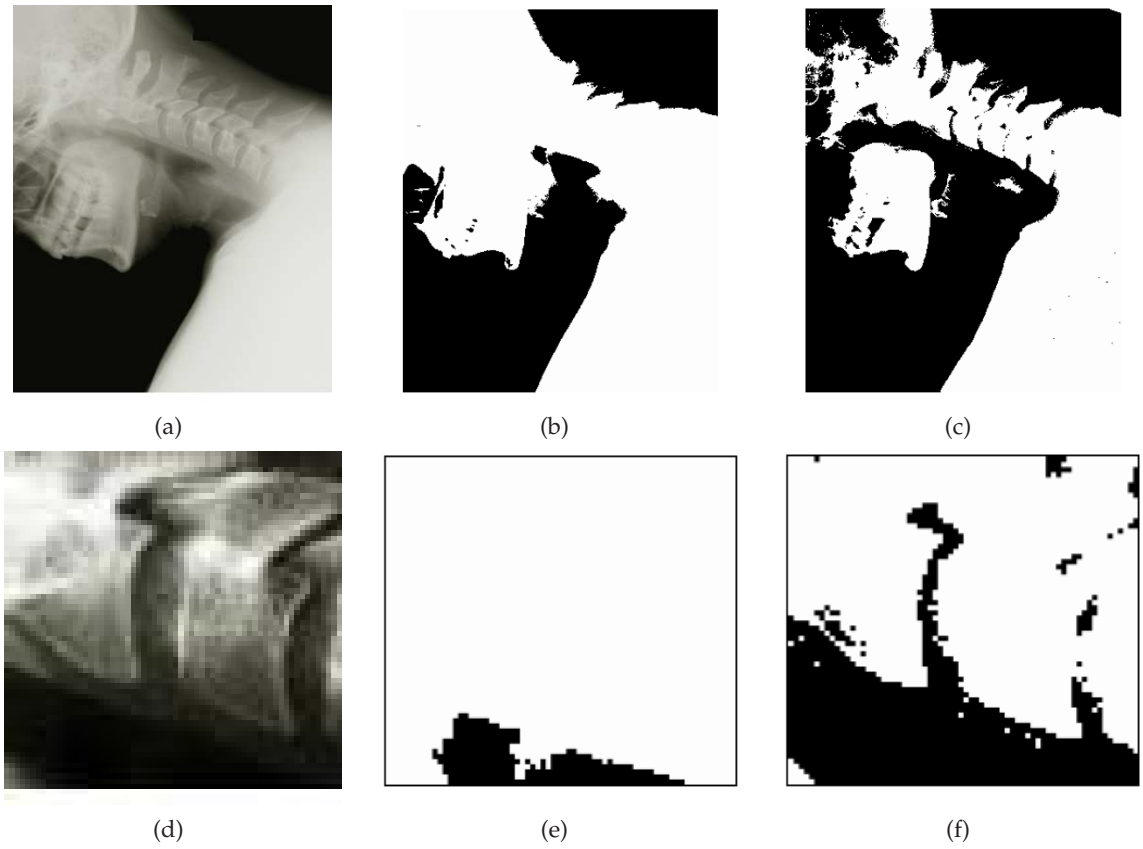


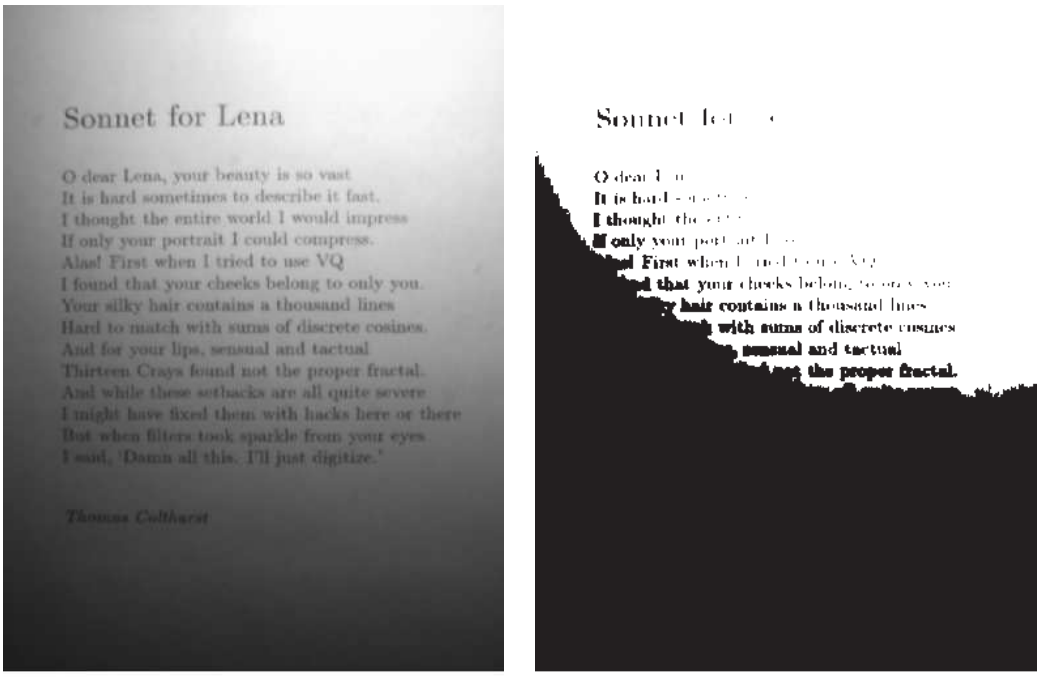
Figura 2.25.: Comparación de resultados entre umbralizado global y el umbralizado adaptativo de Chow y Kaneko. (a) Imagen original. (b) Resultado aplicando un umbral global. (c) Resultado obtenido aplicando la técnica de Chow y Kaneko. (d) Uno de los bloques de 64×64 píxeles en los que se divide la imagen para estimar los umbrales. (e) Resultado de la segmentación en el mismo bloque mediante umbralizado global. (f) Resultado de la segmentación en el bloque mediante la técnica de Chow y Kaneko. Imágenes tomadas de [16]

La figura 2.25 muestra una comparación entre los resultados obtenidos con un umbral global y la técnica de Chow y Kaneko. Los resultados mediante umbralización adaptativa son claramente mejores ya que calcula umbrales basados en subregiones de la imagen.

La principal desventaja del método de Chow y Kaneko es que es muy caro computacionalmente y no resulta práctico para aplicaciones en tiempo real. La alternativa del *umbralizado local* se basa en calcular el umbral para cada píxel estadísticamente usando los valores de intensidad de los píxeles vecinos. El tipo de función estadística a usar depende en gran medida de la imagen

de entrada. Las funciones más comunes y simples de usar son la *media*, la *mediana* y la *media de los valores máximo y mínimo*. Mediante umbrales locales basados en esta estrategia es posible tratar imágenes cuyos histogramas no presentan modos claramente diferenciados.

La figura 2.26(b) muestra el resultado obtenido al tratar de segmentar el texto en una imagen con un fuerte gradiente en su iluminación. Mediante un único umbral global el resultado es inaceptable. La iluminación de la imagen de entrada altera el histograma global de forma que es imposible separarlo mediante un umbral global.



(a) (b)

Figura 2.26.: La segmentación de imágenes con alto gradiente en su iluminación es imposible mediante un único umbral global. (a) Imagen original. (b) Resultado aplicando un único umbral global.

Pueden obtenerse resultados mucho mejores mediante la aproximación de umbralizado local. Recordemos que esta estrategia se basa en la suposición de que es más probable que la iluminación sea uniforme en zonas pequeñas de la imagen. Sin embargo, estas zonas deben ser suficientemente grandes para que contengan píxeles correspondientes al objeto y al fondo. De esta forma, la media (u otro operador estadístico) de las intensidades de los píxeles vecinos al punto que estamos umbralizando en ese momento podrá ser un umbral apropiado para separar el objeto del fondo.

En la figura 2.27(a) puede verse el resultado del umbralizado adaptativo local usando ventanas de tamaño 7×7 para calcular la media y usarla como umbral para cada píxel de la imagen. La segmentación resultante de la figura 2.27(a) es mucho más satisfactoria que la que obtuvimos usando un umbral global pero todavía es deficiente. El defecto principal de esta segmentación es

que, al usar al media, cuando umbralizamos los píxeles que se encuentran en un área de intensidad uniforme (el fondo blanco del papel) el umbral resultante está muy próximo al valor de la intensidad del píxel central. Esto provoca que los píxeles que se encuentran en los márgenes de la página sean etiquetados como fondo en unas ocasiones y como objeto en otras. Una forma de solucionar este problema sin recurrir a funciones estadísticas más complicadas es desplazar ligeramente el valor del umbral que estamos usando. Esto es, en lugar de usar $T = media$ usaremos $T = media - C$, donde C es una constante. De esta forma, todos los píxeles que se encuentren en un entorno de intensidad uniforme serán etiquetados como fondo. La figura 2.27(b) muestra la segmentación de la misma imagen usando esta simple corrección. La ventana usada alrededor de cada píxel para calcular su umbral es de 7×7 y la corrección es $C = 7$. Como puede verse el resultado es mucho más limpio y la segmentación del texto es bastante buena.

El tamaño de las ventanas afecta al proceso de segmentación en dos aspectos diferentes: En primer lugar, las ventanas demasiado grandes producirán segmentaciones de peor calidad ya que el efecto de la iluminación será mayor sobre ellas. Por otra parte, las ventanas grandes son más caras computacionalmente ya que debe calcularse la función estadística para toda el área de la ventana en todos los píxeles de la imagen. En la figura 2.27(c) se han usado ventanas de 75×75 y un desplazamiento del umbral $C = 10$. El resultado es claramente peor que en el caso anterior.

La función estadística a escoger para calcular el umbral depende de la imagen de entrada. Por simplicidad, la más común es la media, que además funciona bastante bien en muchos tipos de imágenes. También la mediana es usada a menudo en aquellos casos en los que se desea que los píxeles con intensidades alejadas de la media (*outliers*) no desvirtúen el umbral calculado. La figura 2.27(d) muestra el resultado de la segmentación con ventanas de tamaño 7×7 y la mediana como función estadística para calcular el umbral. En este caso resulta mucho mejor solución el uso de la media.

En la sección anterior vimos un ejemplo (ver figura 2.23) donde el umbralizado global fracasaba al intentar segmentar una pieza de color oscuro sobre un fondo más claro a causa de la iluminación de la imagen. Hemos visto como el umbralizado local es capaz de resolver este tipo de problemas a la hora de segmentar texto. El éxito de esta técnica sobre la imagen de texto se debe, en gran medida, a que es fácil escoger un tamaño de ventana donde la iluminación es uniforme y al mismo tiempo se abarcan píxeles correspondientes al objeto y al fondo. En el caso de la imagen de la figura 2.23 no resulta tan simple realizar la segmentación debido a la naturaleza de la imagen. Como el objeto a segmentar es relativamente grande respecto al tamaño de la imagen, es difícil aplicar una ventana a la que no afecte la iluminación y sea, al mismo tiempo, lo suficientemente grande para contener píxeles pertenecientes al fondo y al objeto.

En la figura 2.28 podemos ver cómo responde la umbralización local frente a este tipo de imágenes. El resultado de la segmentación es mucho mejor que el obtenido en las figuras 2.23(c) y

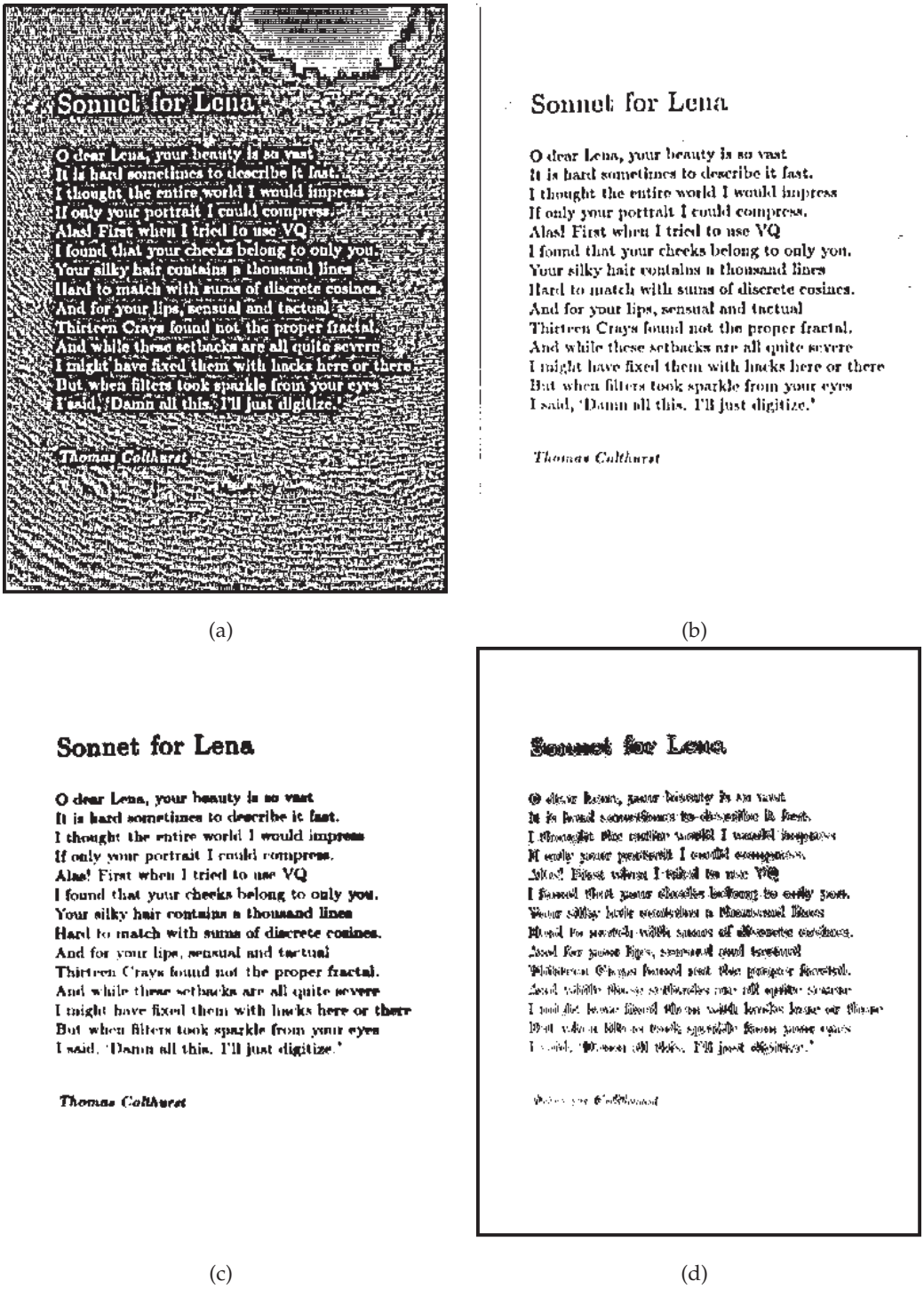
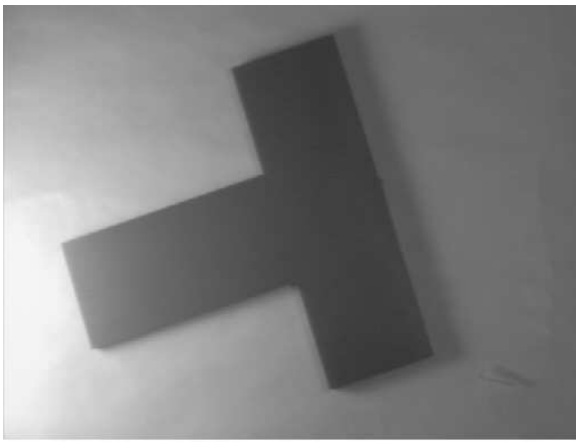
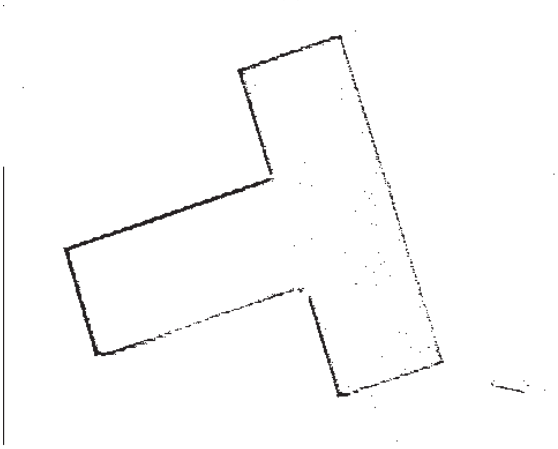


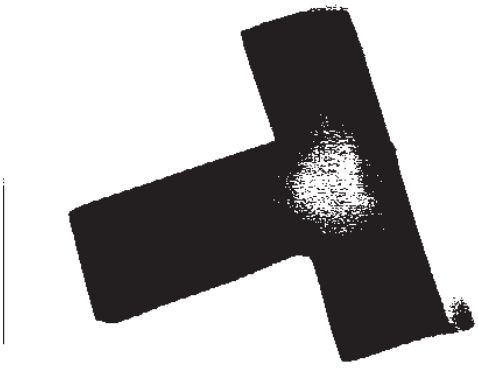
Figura 2.27.: Resultados de la segmentación de texto mediante umbralizado adaptativo local. (a) Media de los 7×7 vecinos. (b) Media de los 7×7 vecinos desplazada con $C = 7$. (c) Media de los 75×75 vecinos desplazada con $C = 10$. (d) Mediana de los 7×7 vecinos desplazada con $C = 4$.



(a)



(b)



(c)

Figura 2.28.: Segmentación de un objeto relativamente grande mediante umbralización local. (a) Imagen original. (b) Segmentación con $T = media - 4$ y ventanas de 7×7. (c) Segmentación con $T = media - 8$ y ventanas de 140×140.

2.23(d) pero la segmentación no es, ni mucho menos, perfecta. La figura 2.28(b) ha sido segmentada con un umbral local $T = media - C$ con $C = 4$ y ventanas de tamaño 7×7. Nótese que todos los píxeles que están dentro del objeto pero no tienen píxeles vecinos diferentes (aquellos que no están próximos al borde) son etiquetados como fondo. Esta imagen recuerda bastante a las obtenidas mediante detección de bordes, sin embargo, no era ése nuestro objetivo en este caso. En la figura 2.28(c) se ha usado una ventana de 140×140 para calcular un umbral local $T = media - C$ con $C = 8$. El exagerado tamaño de la ventana es un intento de lograr que abarque píxeles del objeto y del fondo. El resultado tampoco es satisfactorio a causa del efecto de las sombras y la iluminación. Además, los píxeles del centro de la pieza siguen siendo etiquetados como fondo. En esta imagen, una aproximación más sofisticada como la de Chow y Kaneko sería más apropiada.

2.4.3. Umbral óptimo

Supongamos que la imagen contiene únicamente dos niveles de gris predominantes. El histograma de esta imagen se puede considerar una estimación de la función densidad de probabilidad de nivel de gris, $p(z)$. Esta función será la mezcla de dos funciones unimodales, una para la región clara y otra para la región oscura de la imagen. Además, los pesos de estas densidades van a ser proporcionales al área de cada modo. Si se conoce o se supone la forma de las funciones de densidad de cada modo, se puede determinar el umbral óptimo (en términos de mínimo error) para la imagen a segmentar. Vamos a suponer que la imagen contiene dos valores de nivel de gris combinados con ruido aditivo Gaussiano. La función densidad de probabilidad de primer orden del nivel de gris de la imagen vendrá dado por

$$p(z) = P_1 p_1(z) + P_2 p_2(z) \ , \tag{2.19}$$

donde, en el caso Gaussiano se tiene

$$p(z) = \frac{P_1}{\sqrt{2\pi}\sigma_1} \exp\left[-\frac{(z-\mu_1)^2}{2\sigma_1^2}\right] + \frac{P_2}{\sqrt{2\pi}\sigma_2} \exp\left[-\frac{(z-\mu_2)^2}{2\sigma_2^2}\right] \ , \tag{2.20}$$

donde μ_1 y μ_2 son los valores medios de nivel de gris, σ_1 y σ_2 las desviaciones estándar en torno a la media y P_1 y P_2 las probabilidades *a priori* de los dos niveles de gris. Se debe satisfacer la restricción

$$P_1 + P_2 = 1 \ , \tag{2.21}$$

de forma, que la densidad $p(z)$ tiene cinco parámetros desconocidos. Si se conocen estos cinco parámetros se puede determinar el umbral óptimo de forma sencilla. Supongamos que la región oscura corresponde al fondo de la imagen, y las regiones claras a objetos. En este caso, $\mu_1 < \mu_2$, y se debe definir un umbral T de forma que los píxeles con nivel de gris menores que T se puedan considerar como píxeles de fondo y los píxeles con nivel de gris mayores que T píxel objeto. La probabilidad de clasificar erróneamente un píxel de objeto como de fondo es

$$E_1 = \int_{-\infty}^{T} p_2(z) dz \ . \tag{2.22}$$

De forma similar, la probabilidad de clasificar erróneamente un píxel de fondo como objeto es

$$E_2 = \int_T^{-\infty} p_1(z)dz \ . \tag{2.23}$$

Entonces la probabilidad total de error será

$$E(T) = P_2 E_1(T) + P_1 E_2(T) \ . \tag{2.24}$$

Para encontrar el valor de T que da lugar a una probabilidad de error de clasificación mínima, podemos derivar la ecuación 2.24 con respecto a T e igualar a cero. Entonces

$$P_1 p_1(T) = P_2 p_2(T) \ . \tag{2.25}$$

En el caso Gaussiano, tomando logaritmos, la ecuación 2.25 resulta en la ecuación de segundo grado

$$AT^2 + BT + C = 0 \ , \tag{2.26}$$

donde

$$A = \sigma_1^2 + \sigma_2^2 \ , \tag{2.27}$$

$$B = 2(\mu_1 \sigma_2^2 - \mu_2 \sigma_1^2) \ , \tag{2.28}$$

$$C = \sigma_1^2 \mu_2^2 - \sigma_2^2 \mu_1^2 + 2\sigma_1^2 \sigma_2^2 \ln \frac{\sigma_2 P_1}{\sigma_1 P_2} \ . \tag{2.29}$$

De las dos soluciones una corresponderá a un máximo local y otra a un mínimo local. Es necesario sustituir ambos resultados en la ecuación 2.24 y descartar la que de lugar a un error mayor, quedándonos con la otra solución como la válida.

En el caso de que las varianzas sean iguales ($\sigma^2 = \sigma_1^2 = \sigma_2^2$) la ecuación 2.26 tiene una única solución que corresponde en este caso al error mínimo. Si las probabilidades a priori son iguales, $P_1 = P_2$, el umbral óptimo viene dado por el valor medio de las medias. Esto mismo se cumple para el caso $\sigma = 0$.

En el caso de que las densidades de probabilidad sigan otras leyes conocidas, como la Rayleigh o la log-normal, se puede aplicar este mismo procedimiento para la determinación del umbral óptimo.

2.4.4. Umbrales basados en varias variables. *Clustering*

Hasta ahora hemos considerado la determinación de umbrales para una única variable: el nivel de gris de la imagen. En algunos casos, se dispone de varias variables que caracterizan cada píxel de la imagen. Un ejemplo muy claro son las imágenes en color, donde se dispone de las componentes RGB para formar la imagen compuesta en color. En este caso, cada píxel se caracteriza por tres variables, y se puede construir un histograma en tres dimensiones. El concepto de segmentación mediante umbral en este caso da lugar el empleo de técnicas de clasificación o *clustering*. Si se han encontrado K *clusters* significativos en el espacio de las variables, la imagen se puede segmentar asignando a cada píxel una de las K etiquetas a todos aquellos píxeles cercanos al *cluster* correspondiente a esa etiqueta. La complejidad de esta clasificación depende del número de variables y del número de *clusters*. Además si se desconoce el número de *clusters* el problema se complica aun más. En el caso particular de que se conozca el número de *clusters* se puede emplear el algoritmo LBG o K-medias. Este algoritmo permite determinar iterativamente el centroide de cada uno de los K *clusters* y la partición del espacio de los parámetros en K zonas, una para cada *cluster*. Partiendo de una estimación inicial para los centroides, se clasifican los datos según esos centroides. A partir de los datos clasificados para cada *cluster* se vuelve a determinar el centroide para ese *cluster*. Esto se repite hasta que la posición de los centroides de los *clusters* y la clasificación de los datos no cambie apreciablemente. La imagen segmentada viene dada entonces por las etiquetas correspondientes a la última clasificación.

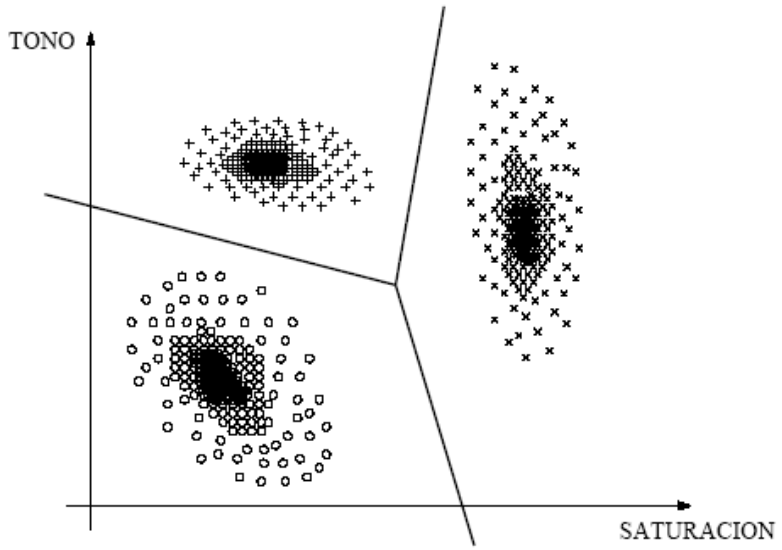


Figura 2.29.: Ejemplo de clustering para dos variables y tres clusters.

Es particularmente importante la clasificación de imágenes en color basada en tono y saturación. Estas propiedades de color son particularmente importantes, a diferencia de los canales

RGB, porque son equivalentes a como se perciben los colores en el sistema visual humano. En este caso tenemos dos variables y se pueden emplear técnicas de clasificación o *clustering* en dos dimensiones. En la figura 2.29 se puede ver un ejemplo para tres clusters junto con las regiones de clasificación (umbrales en dos dimensiones).

2.5. Técnicas basadas en regiones

El objetivo de toda segmentación es dividir la imagen en regiones. En las secciones precedentes hemos visto estrategias que abordaban el problema buscando las fronteras entre las regiones basándose en discontinuidades de intensidad. También hemos visto formas de segmentación basadas en umbrales que pueden ser aplicados a diferentes propiedades de los píxeles como la intensidad o el color. En esta sección estudiaremos técnicas de segmentación que tratan de crear las regiones de la segmentación directamente.

Para evitar ambigüedades y clarificar las explicaciones subsiguientes es útil aclarar previamente la terminología y formulación que usaremos en esta sección: Consideraremos la imagen a segmentar R como una región que cubra toda la imagen. De forma que el proceso de segmentación que vamos a seguir consiste en crear n subregiones R_1,R_2,....,R_n a partir de la región R inicial. Estas subregiones deberán cumplir una serie de condiciones:

1. $\cup_{i=1}^{n} R_i = R$.
2. R_i es una región conectada, $i = 1, 2, ..., n$.
3. $R_i \cap R_j = \phi$ para todo i y $j, i \neq j$.
4. Se cumple $P(R_i)$ para todo $i = 1, 2, ..., n$.
5. No se cumple $P(R_i \cup R_j)$ para $i \neq j$.

Donde $P(R_i)$ es un predicado lógico sobre los puntos del conjunto R_i y ϕ representa el conjunto vacío.

La condición (1) indica que la segmentación debe ser completa, es decir, cada píxel debe pertenecer a una región. La segunda condición exige que los puntos pertenecientes a una región deben estar conectados (ver conectividad en la sección 2.1). La tercera condición indica que las regiones no deben compartir píxeles. La cuarta condición hace referencia a las propiedades que deben cumplir los píxeles pertenecientes a una región. Por ejemplo, se cumple $P(R_i)$ si todos los píxeles en R_i tienen la misma intensidad. La última condición expresa la diferencia entre las propiedades de los píxeles pertenecientes a diferentes regiones.

2.5.1. Crecimiento de regiones

Como su propio nombre indica, *crecimiento de regiones* es un procedimiento por el que píxeles o grupos de píxeles son agrupados en regiones de mayor tamaño. La forma más simple de esta técnica es conocida como *suma de píxel*. Esta estrategia comienza con un determinado número n de píxeles *semilla* que forman las regiones iniciales R_1,R_2,...,R_n. A partir de los píxeles semilla las regiones empiezan a crecer anexionándose los píxeles vecinos que cumplen algún criterio de similitud preestablecido (nivel de gris, textura, color...).

Para ilustrar este proceso podemos echar un vistazo a la figura 2.30(a), donde los números en las casillas representan el nivel de gris de cada píxel. Los píxeles con coordenadas (3,2) y (3,4) serán las semillas. Como tenemos dos puntos de partida, tenemos dos regiones iniciales: R_1 asociada con la semilla en (3,2) y R_2 asociada con la semilla (3,4). El criterio P que usaremos para decidir si un píxel es incluido o no en una región será el siguiente: la diferencia en valor absoluto entre el nivel de gris del píxel evaluado y el nivel de gris del píxel semilla deberá ser menor que un umbral T. Cualquier píxel que satisfaga esta condición para las dos regiones será asignado, de forma arbitraria, a la región R_1. La figura 2.30(b) muestra el resultado obtenido usando un umbral $T = 3$. En este caso, la segmentación ha resultado en dos regiones, donde los puntos pertenecientes a R_1 están etiquetados como *a* y los puntos pertenecientes a la región R_2 están etiquetados con la letra *b*. Nótese que cualquier posición de las semillas para cualquiera de las dos regiones hubiera dado como resultado la misma segmentación. En la figura 2.30(c) se muestra el resultado de la segmentación para $T = 8$. En este caso la región R_1 a cubierto el total de la imagen ya que todos los píxeles que cumplen la condición son asignados a R_1.

De la explicación anterior pueden deducirse algunas de las dificultades fundamentales que presenta la técnica de crecimiento de regiones. Uno de los problemas es la elección adecuada del número y la posición de las semillas iniciales, ya que las propiedades de los píxeles iniciales determinarán el resultado final. Interesa, por lo tanto, que las semillas iniciales sean representativas de una región de interés en la imagen. Otro problema evidente es la elección del criterio que usará el algoritmo para agregar nuevos píxeles a las regiones ya existentes. La elección de una o más semillas iniciales está influenciada a menudo por la naturaleza del problema. Por ejemplo, en las aplicaciones militares basadas en imágenes de infrarrojos, los objetivos de interés suelen ser aquellos que tienen una temperatura mayor y, por lo tanto, aparecen en la imagen con niveles de brillo más altos. En este caso un buen criterio de elección para el píxel semilla es usar aquellos píxeles con mayor nivel de brillo. Cuando no disponemos de información a priori sobre la naturaleza de la imagen el método a seguir es evaluar todos los píxeles de la imagen según las propiedades que vayamos a usar durante el proceso de crecimiento (nivel de gris, texturas, color...). Si el resultado de este análisis muestra ciertas agrupaciones o *clusters* de valores, una buena elección para la posición de las semillas serán los centroides de estos *clusters*. En el ejemplo que ilustra la figura

	1	2	3	4	5
1	0	0	5	6	7
2	1	1	5	8	7
3	0	1	6	7	7
4	2	0	7	6	6
5	0	1	5	6	5

(a)

a	a	b	b	b
a	a	b	b	b
a	a	b	b	b
a	a	b	b	b
a	a	b	b	b

(b)

a	a	a	a	a
a	a	a	a	a
a	a	a	a	a
a	a	a	a	a
a	a	a	a	a

(c)

Figura 2.30.: Ejemplo de la técnica de crecimiento de regiones. (a) Imagen original con los valores de gris de los píxeles. Los píxeles semilla están subrayados. (b) Segmentación usando $T = 3$. (c) Segmentación usando $T = 8$.

2.30, una representación del histograma de la imagen mostraría que predominan los puntos con intensidades de 1 y 7.

La elección de un criterio de similitud no sólo depende del problema que estemos abordando sino también del tipo de datos del que disponemos. En aplicaciones de análisis geográfico basadas en imágenes de satélite es muy importante el uso del color. La resolución de problemas de este tipo sería mucho más difícil con imágenes monocromáticas. Desafortunadamente, en la mayoría de los casos no contamos con datos multiespectrales de las imágenes y, típicamente, el análisis mediante técnicas basadas en regiones ha de llevarse a cabo usando descriptores basados propiedades espaciales y de intensidad a partir de una única imagen.

Por otra parte, el uso exclusivo de los descriptores para la segmentación basada en regiones puede llevar a resultados erróneos si no se usa información sobre la conectividad durante el proceso de crecimiento. El hecho de agrupar los píxeles presentes en una imagen según sus descriptores (por ejemplo intensidad) independientemente de la posición que ocupen unos respecto a otros dentro de la imagen no producirá ningún resultado satisfactorio.

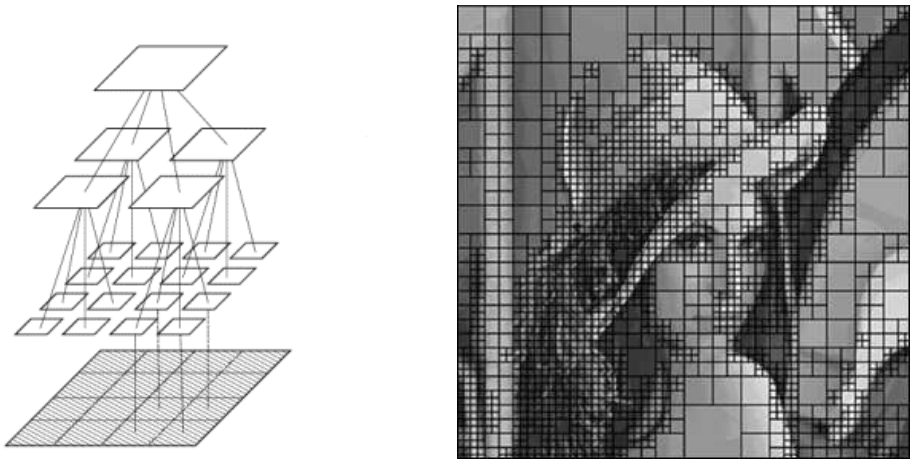
Otro problema es la formulación de un criterio de parada para el algoritmo. En principio, una región debería dejar de crecer cuando no existan más píxeles que satisfagan su criterio de similitud. Sin embargo, a menudo es útil añadir un criterio adicional que tenga en cuenta el tamaño de la región o su forma. Los descriptores como intensidad o textura no tienen en cuenta la "historia" de la región de forma que, en muchas ocasiones, es recomendable comparar el descriptor del píxel candidato a unirse a la región con el descriptor de la región completa (por ejemplo, la intensidad del píxel candidato y la intensidad media de todos los píxeles que forman la región). El uso de estos criterios adicionales está siempre condicionado por la información a priori disponible.

2.5.2. División y fusión de regiones

En la anterior sección hemos visto el proceso por el cual las regiones crecen a partir de unos cuantos píxeles semilla. Una aproximación alternativa consiste en dividir la imagen inicialmente en un grupo de regiones de forma arbitraria y luego fundir o dividir estas regiones tratando de cumplir las condiciones expuestas en la sección 2.5. A continuación se explica el funcionamiento de un algoritmo de fusión y división de regiones que trata de satisfacer estas condiciones trabajando de forma iterativa.

Sea R una región que abarca toda la imagen y P un predicado lógico como el descrito en la sección 2.5. Considerando una imagen de forma cuadrada, una aproximación para segmentar R puede consistir en dividir y subdividir la imagen sucesivamente en cuadrantes cada vez más

pequeños de forma que para toda región R_i se cumpla $P(R_i)$ (esto es, todos los píxeles dentro de R_i son homogéneos). Es decir, si $P(R)$ no se cumple, se divide R en cuadrantes. De la misma forma, si P no se cumple para algún cuadrante, se subdivide ese cuadrante y así sucesivamente. Esta técnica en particular suele representarse mediante gráficos en forma de árbol llamados *quadtree* (cada nodo tiene siempre cuatro descendientes). En la figura 2.31 podemos ver un ejemplo de este tipo de representación y el resultado obtenido tras aplicar el algoritmo sobre una imagen real.



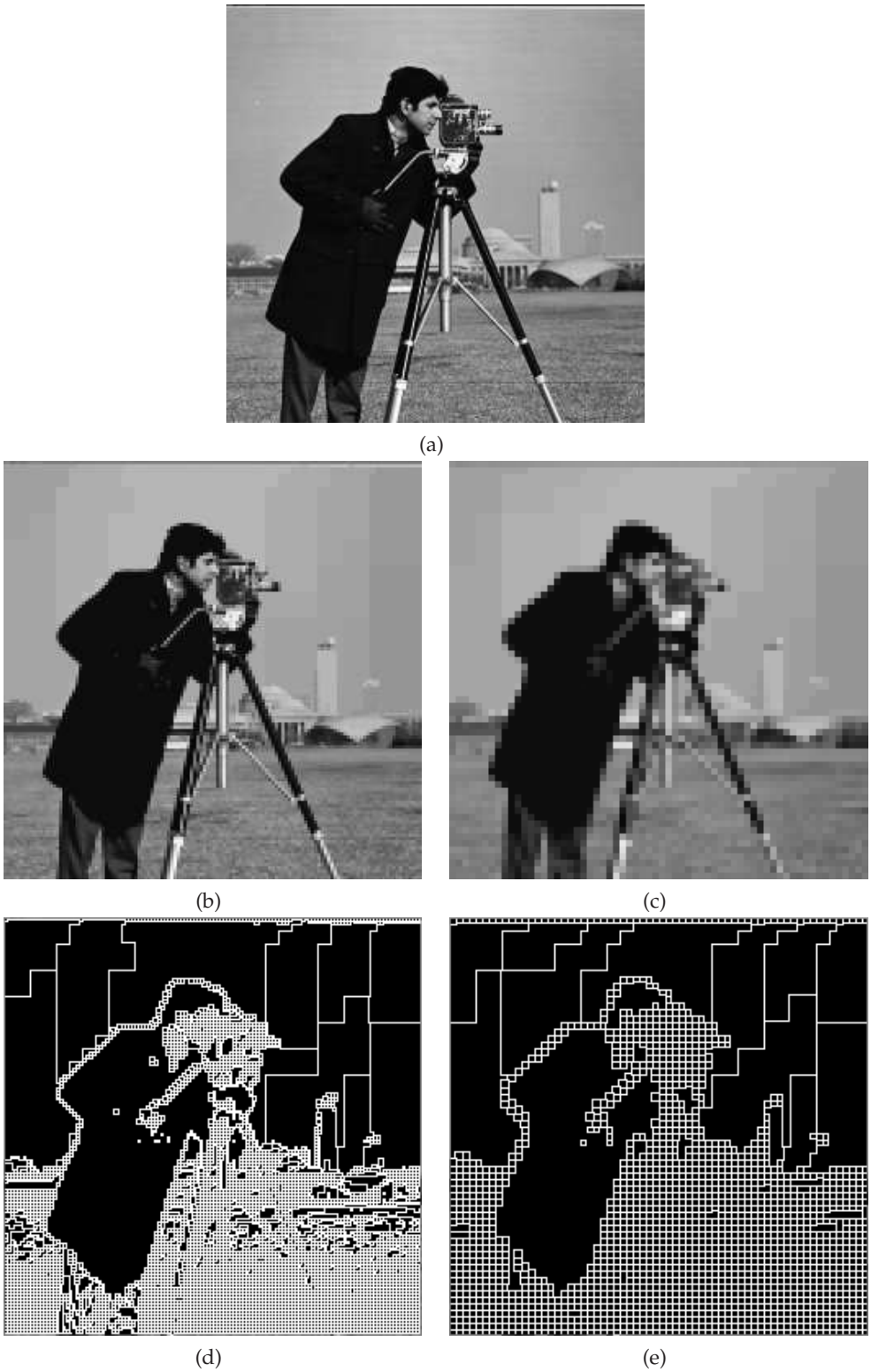
(a) (b)

Figura 2.31.: (a) Representación de la estructura de árbol quadtree.(b) Aplicación de la técnica de división de regiones sobre una imagen.

Si sólo se permite al algoritmo realizar divisiones, es muy probable que existan subdivisiones adyacentes con propiedades idénticas o muy similares. Este problema puede remediarse permitiendo que el algoritmo realice fusiones entre estas regiones similares. Para satisfacer las condiciones expresadas en la sección 2.5 el algoritmo sólo deberá fundir aquellas regiones adyacentes cuyos píxeles satisfagan el predicado P, es decir, dos regiones adyacentes R_j y R_k se fundirán si y sólo si cumplen que $P(R_j \cup R_k)$. Lo descrito anteriormente puede sintetizarse mediante el siguiente algoritmo iterativo:

1. Dividir en 4 cuadrantes cualquier región R_i donde no se cumpla $P(R_i)$.
2. Fundir las regiones adyacentes R_j y R_k que cumplan $P(R_j \cup R_k)$.
3. Detener el algoritmo cuando ya no puedan dividirse o fundirse más regiones.

En la figura 2.32 podemos ver un ejemplo de segmentación de una imagen siguiendo este algoritmo. Las figuras 2.32(b) y 2.32(c) muestran la imagen obtenida tras aplicar el procedimiento de división y fusión de regiones. En la figura 2.32(c) el tamaño mínimo permitido a los cuadrantes es el doble que en el caso (b). Las figuras 2.32(d) y 2.32(e) muestran el resultado de la segmentación para cada caso.



(a) (b) (c) (d) (e)

Figura 2.32.: Segmentación mediante división y fusión de regiones.(a) Imagen original. (b) Resultado tras dividir la imagen en 7973 regiones y después fundir hasta 5744 regiones. (c) Resultado tras dividir la imagen en 2389 regiones y después fundir hasta 1839 regiones. (d) Segmentación obtenida a partir de (b). (e) Segmentación obtenida a partir de (c).

Las representaciones basadas en regiones (como los *quadtree*) ofrecen una forma de realizar un primer nivel de abstracción y reducir así el número de elementos a procesar con respecto a la representación clásica basada en el píxel. Las representaciones en forma de árbol son muy utilizadas ya que posibilitan la representación de la imagen de forma jerárquica y permiten aplicar sobre ellas técnicas de proceso eficientes y complejas.

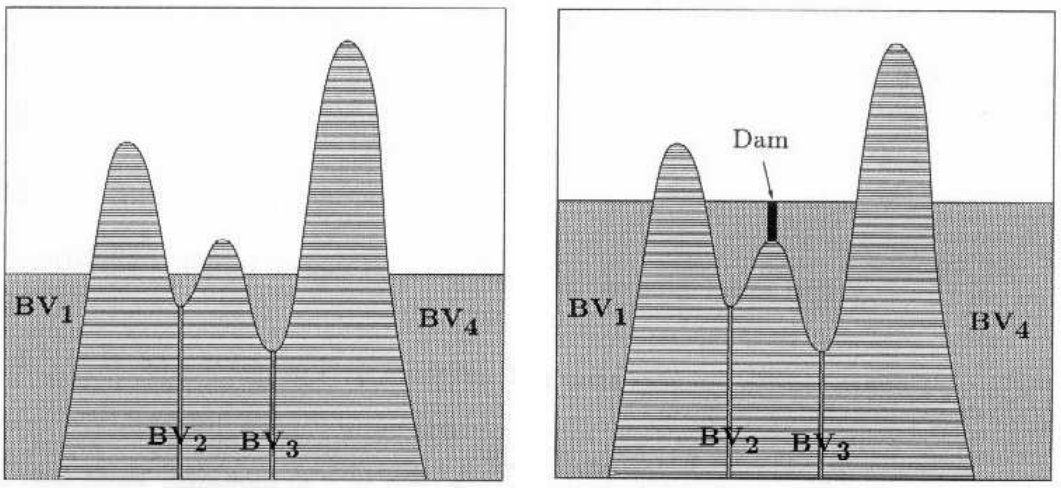
Las estrategias de procesado se basan en técnicas de poda. Estas técnicas eliminan algunas ramas del árbol (funden regiones) basándose en análisis realizados sobre los nodos. El resultado obtenido tras aplicar las técnicas de poda sobre los diagramas de árbol es una imagen filtrada o segmentada.

2.5.3. Segmentación morfológica: *Watersheds*

Otra forma de segmentación basada en regiones es conocida como *watersheds* (en inglés, línea divisoria de aguas). La primera proposición de este tipo fue introducida por Meyer y Beucher [12] en 1998. Esta estrategia se inspira en la geología, considerando a la imagen digital como una superficie cuyo relieve corresponde a los niveles de grises de los píxeles que la forman. Si representamos una imagen siguiendo este criterio tendremos como resultado una superficie con valles y crestas. Los valles de la superficie se corresponderán con mínimos regionales y las cimas identificarán máximos regionales. Las regiones serán entonces las zonas aproximadamente planas de esta superficie y estarán separadas ente sí por zonas con altas pendientes.

Pero, ¿todo esto qué tiene que ver con las líneas divisorias de agua? La técnica de *watersheds* debe su nombre al proceso que sigue a continuación: Supongamos un relieve topográfico como el de la figura 2.33(a) que empieza a inundarse de agua desde el fondo de las cuencas hacia arriba y con velocidad vertical constante. Llegará un momento en el que el agua que está llenando dos o más cuencas contiguas superará la cima que separa los dos lagos. Si en el punto donde las dos masas de agua se unen construimos una presa para evitar que esto ocurra, cuando el agua haya cubierto toda la superficie topográfica, lo único que sobresaldrá del agua serán las presas. Estas presas (líneas divisorias) definen el *watershed* de la imagen. En la figura 2.33 se ilustra este proceso el resultado final son varios lagos separados por las presas y en cada uno de ellos se encuentra un único mínimo.

Existe una forma alternativa de aplicar esta técnica. Consiste en enlazar un píxel con su vecino de más bajo nivel de gris. A continuación enlazamos ese píxel con el siguiente vecino de más bajo nivel de gris y así sucesivamente hasta encontrar un mínimo regional. El área que contiene el grupo de píxeles que bajan hasta un mínimo común forma un lago del *watershed*. Esta alternativa se conoce como *tobogganing* porque simula el agua deslizándose hacia abajo en lugar de inundar las cuencas de forma ascendente.



(a) (b)

Figura 2.33.: Proceso de inundación de watersheds (a) El proceso de inundación ha comenzado pero todavía no hemos construido la presa.(b) Instantes después ya hemos construido la presa para evitar que se fundan los lagos. Los mínimos de la imagen y los diferentes lagos están etiquetados.

La aplicación de la técnica de *watersheds* sobre una imagen real presenta algunas dificultades. En primer lugar, la segmentación mediante esta técnica sobre una imagen sin preprocesar puede producir resultados no satisfactorios en función de la naturaleza de la imagen. Por esta razón suele aplicarse la técnica de *watersheds* sobre el gradiente de la imagen. En la figura 2.34(b) podemos ver la diferencia entre las superficies topológicas de una imagen y las de su gradiente. Como se puede apreciar, la superficie correspondiente al gradiente de la imagen tiene forma de cráter (ver sección 2.3.1) y resulta más apta para la técnica de inundación. Por esta razón, en la gran mayoría de los casos se aplica el gradiente antes de usar *watersheds*.

En imágenes ligeramente más complejas que la de la figura 2.34 surge el verdadero problema de esta técnica. Como vimos en la sección 2.3.1, el operador gradiente resulta especialmente sensible al ruido. Esto resulta muy perjudicial para la aplicación de esta estrategia ya que aparecen múltiples mínimos regionales que provocan la sobresegmentación de la imagen. Podemos ver un ejemplo de este efecto en la figura 2.35(b), en la que se ha aplicado *watersheds* directamente sobre el gradiente de la imagen. Este efecto puede reducirse mediante el uso de otros operadores derivativos con efecto de suavizado (sección 2.3.2) pero se obtienen mejores resultados imponiendo la posición de los mínimos regionales. Estas "marcas" pueden obtenerse hallando los mínimos de la imagen antes de calcular su gradiente. Es habitual incluir también un marcador para el fondo de la imagen. El número de regiones obtenidas mediante esta técnica coincide siempre con el número de mínimos marcados de forma que puede controlarse el resultado final de la segmentación.

En la figura 2.35(c) vemos el conjunto de marcadores definido a partir de la imagen original. El resultado (figura 2.35) de la segmentación en este caso es claramente mejor.

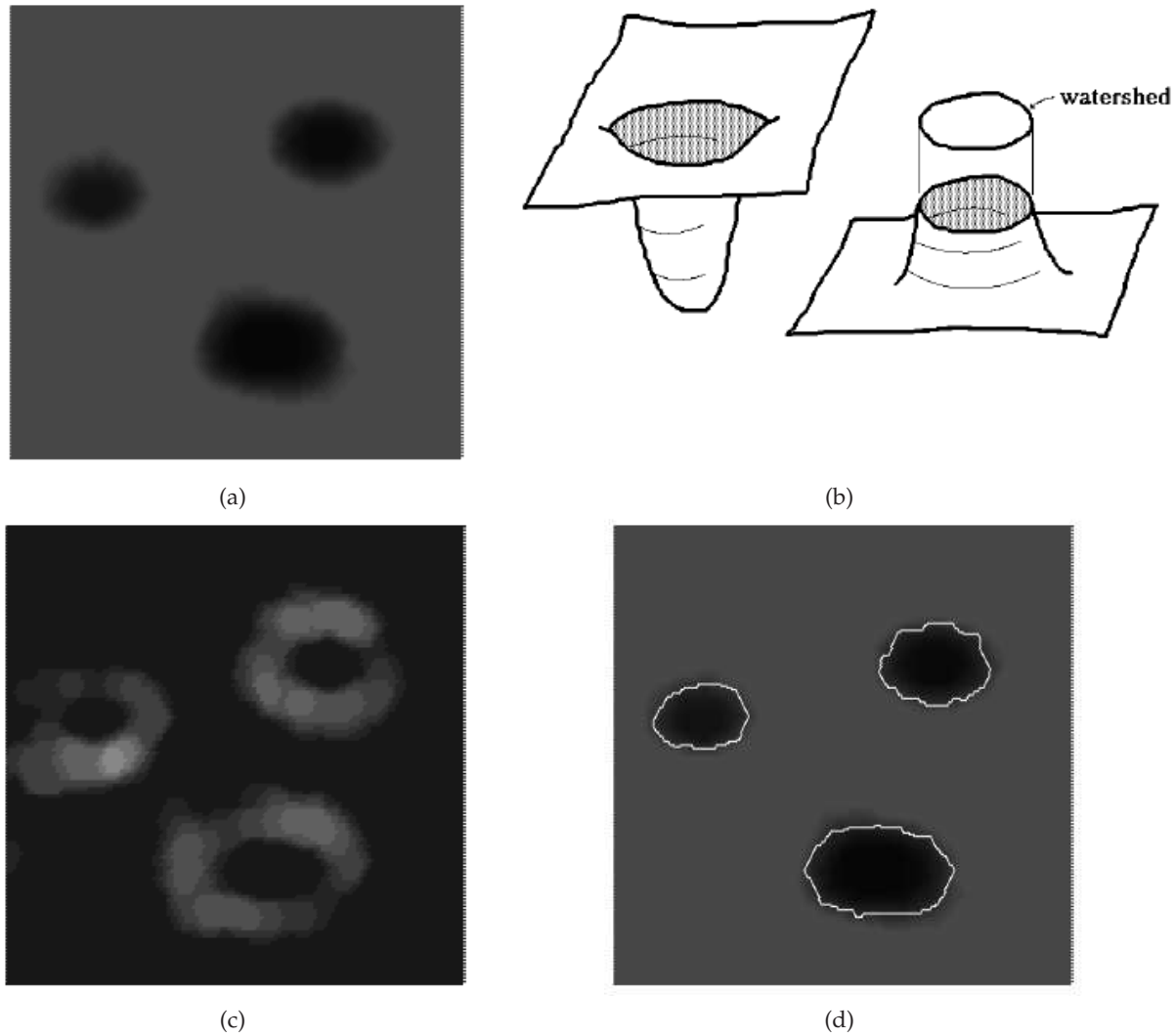
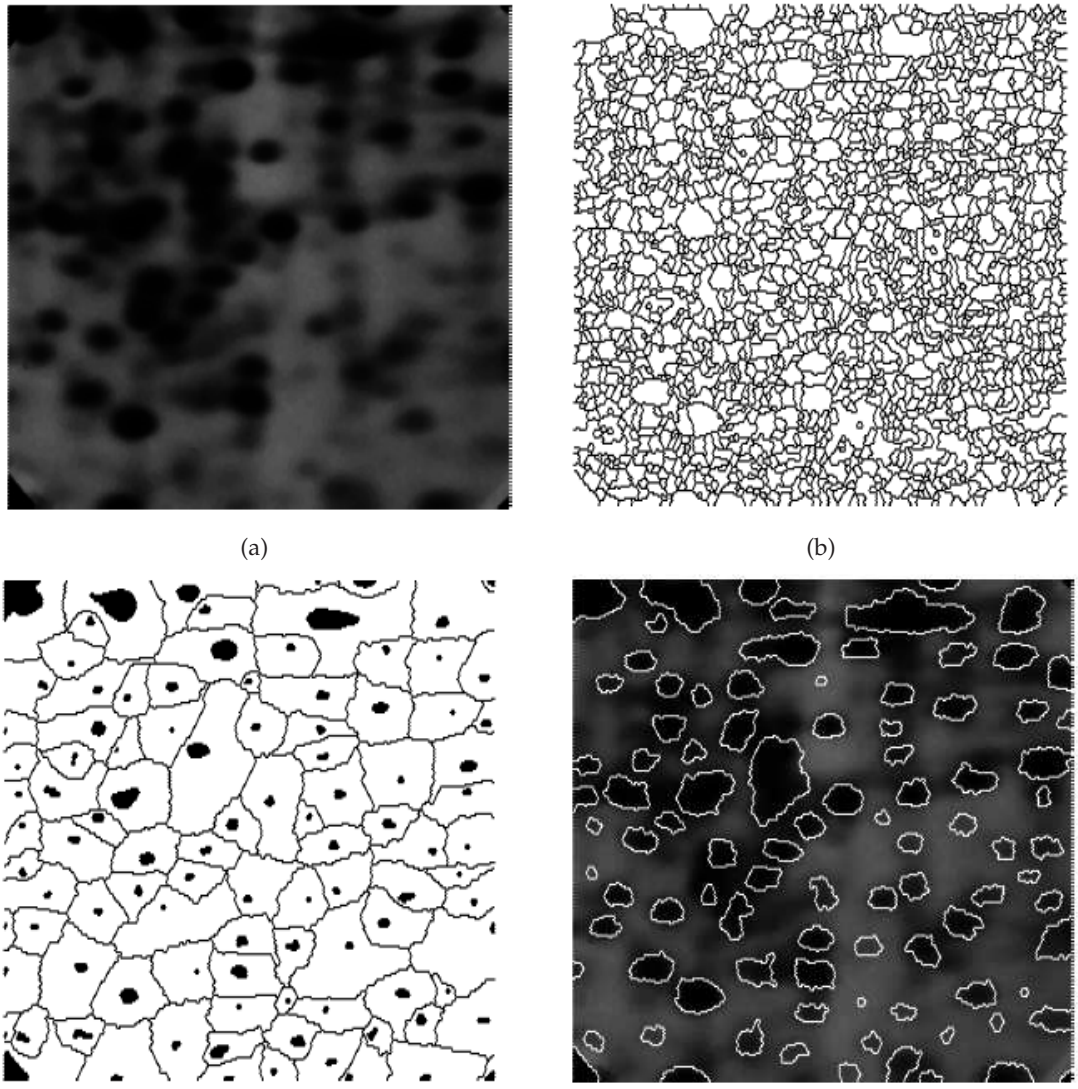


Figura 2.34.: Segmentación de una imagen mediante watersheds. (a) Imagen original (b) Representación del relieve de una porción de la imagen original (izquierda) y representación del gradiente de la misma porción (derecha). (c) Gradiente de la imagen original (d) Watershed de la imagen gradiente superpuesto sobre la imagen original. Imágenes tomadas de [15]



(a) (b)

(c) (d)

Figura 2.35.: Sobresegmentación con watersheds. (a) Imagen original (b) Resultado tras aplicar watersheds sobre el gradiente de la imagen. (c) Obtención de los mínimos a partir de la imagen original (d) Segmentación usando los mínimos marcados. Imágenes tomadas de [15].

La forma habitual de determinar el número y la posición aproximada de los mínimos que nos interesan se suele calcular mediante una operación morfológica conocida como *reconstrucción geodésica* que se realiza sobre la imagen antes de comenzar la inundación. Esta operación impone los mínimos mediante una función externa (función marcadora) de forma que los mínimos regionales que no interesan son rellenados (o erosionados los máximos locales relativos a ellos) mediante reconstrucción morfológica convirtiéndose en zonas no mínimas. Así se consigue que sólo las zonas marcadas produzcan una región al finalizar el proceso.

Hay dos causas principales de sobresegmentación cuando usamos *watersheds*: El ruido y la escasa dinámica de algunas imágenes. En la figura 2.36(a) vemos que aparecen mínimos locales indeseados debidos al ruido. Si tenemos suficiente información a priori de la imagen y sabemos lo que estamos buscando podemos implementar una función morfológica que se encargue de erosionar los picos producidos por el ruido en el relieve de la imagen. De esta forma los mínimos relativos desaparecen y no generarán regiones indeseadas (figura 2.36(b)).

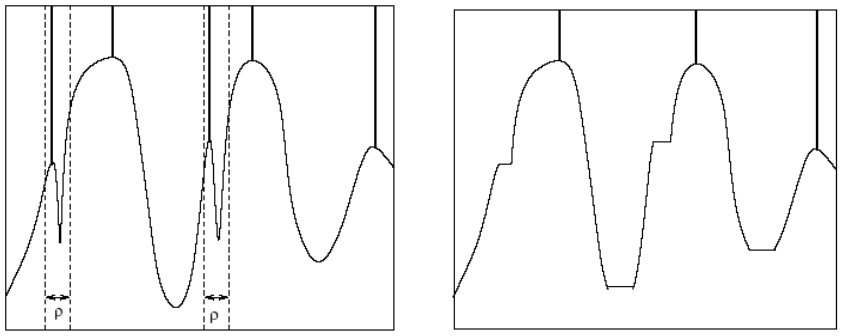


Figura 2.36.: Ejemplo de reconstrucción geodésica en una imagen ruidosa. (a) El ruido ha producido picos que generarán regiones no deseadas. (b) Los lagos con anchura inferior a ρ se filtran morfológicamente hasta que desaparecen los mínimos relativos a ellos.

La figura 2.37(a) ilustra el caso de una imagen con baja dinámica. Esto es, las diferencias entre los niveles de grises de los píxeles son muy sutiles. Si conocemos la naturaleza de la imagen y sabemos cuál es nuestro objetivo pueden descartarse los lagos con una profundidad menor que la especificada por la función marcadora. Esta situación se representa en la figura 2.37(b). Inicialmente se producen cuatro líneas divisorias que, tras imponer la condición de profundidad (h), se reducen a dos.

La segmentación de imágenes mediante *watersheds* produce buenos resultados para una gran variedad de imágenes, incluso en aquellas con poco contraste. El éxito de esta técnica se debe, en gran medida, a que su principio de funcionamiento es muy intuitivo. Además, es fácil de implementar y rápida de ejecutar. Al ser una técnica basada en regiones, produce una segmentación directa de la imagen sin necesidad de ningún postprocesado adicional para unir bordes inconexos.

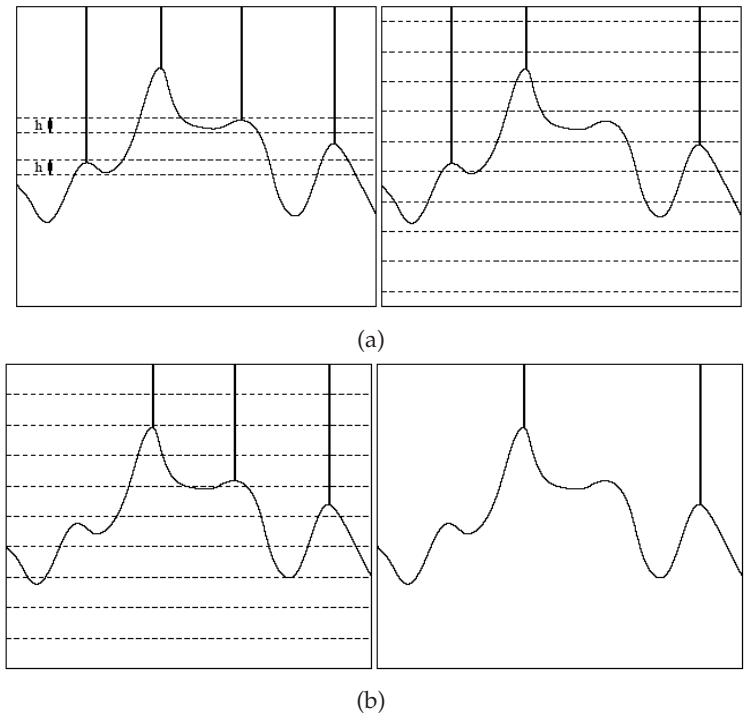


Figura 2.37.: Reconstrucción geodésica en una imagen con baja dinámica. (a) La función marcadora detecta los lagos de profundidad menor que h. (b) El resultado final sólo deja las watersheds deseadas.

Otra característica importante de la segmentación mediante *watersheds* es que el proceso se divide en etapas. En primer lugar hay que seleccionar lo que queremos segmentar (función marcadora) y en segundo lugar se realiza la segmentación. Esto es una ventaja siempre que tengamos suficiente información a priori acerca de la imagen pero puede llevar a resultados desastrosos en el caso contrario.

2.6. Técnicas basadas en modelos deformables

Los modelos deformables son curvas o superficies definidas dentro del dominio de una imagen que pueden deformarse en función de unas fuerzas aplicadas sobre ellas. Estas fuerzas pueden ser de dos tipos:

- Fuerzas internas: definidas por la curva o superficie misma.
- Fuerzas externas: que se calculan en función de los datos de la imagen.

Las fuerzas internas se diseñan para mantener el modelo en una forma suave durante el proceso de deformación, evitando así que las fronteras definidas por las curvas aparezcan ruidosas. Las fuerzas externas se definen de forma que arrastren el modelo hacia los bordes de uno o varios objetos que son el objetivo de la segmentación. Si se tiene suficiente información pueden definirse ambos tipos de fuerzas de forma que la curva o superficie converja hacia el objeto a segmentar formando una frontera suave y precisa, consiguiendo así un método de segmentación robusto frente al ruido. Además, al ser estas curvas definidas mediante ecuaciones matemáticas totalmente coherentes y consistentes, el resultado de la segmentación puede ser usado en otras aplicaciones independientemente de las características de la imagen (resolución, formato...) y extraer parámetros de interés a partir de esta representación matemática. Esta característica es interesante, por ejemplo, para construir modelos en tres dimensiones a partir de segmentaciones en dos dimensiones. Por otra parte, el hecho de poder expresar estas curvas mediante expresiones matemáticas definidas en el dominio continuo permite lograr precisiones superiores al tamaño del píxel, característica muy interesante en aquellas aplicaciones donde la resolución de la imagen es pobre respecto al tamaño del objeto a segmentar.

Las técnicas basadas en modelos deformables vienen estudiándose desde hace tiempo pero su uso se hizo especialmente popular a partir del trabajo de Terzopoulos y sus colaboradores [13] a finales de los ochenta. En la literatura pueden encontrarse multitud de términos para referirse a los modelos deformables: contornos activos, *snakes*, *balloons*, superficies deformables, etc. Todos ellos comparten las características básicas que hemos comentado al principio de esta sección.

Básicamente, existen dos tipos de modelos deformables: paramétricos y geométricos. Los *modelos deformables paramétricos* expresan las curvas en su forma paramétrica explícita durante su deformación. Este tipo de representación permite una interacción directa con el modelo y es muy recomendable para conseguir una segmentación compacta en poco tiempo. Sin embargo, realizar adaptaciones topológicas tales como la división de la curva en dos partes puede resultar complicado con este tipo de representación. Por otra parte, los *modelos deformables geométricos* son ideales para realizar adaptaciones topológicas de este tipo. Estos modelos no realizan la computación de los parámetros hasta que ha finalizado la deformación de forma que es fácil adaptar los cambios topológicos sufridos por el modelo. Al margen de esta diferencia fundamental, ambos modelos siguen principios de funcionamiento similares.

En la siguiente sección nos centramos en el modelo deformable paramétrico más común: los contornos activos o *snakes*.

2.6.1. Contornos deformables (*Snakes*)

De entre los diferentes tipos de modelos deformables existentes, escogemos los contornos deformables (*snakes*) como un buen ejemplo para ilustrar el funcionamiento de este tipo de algoritmos de segmentación. Nos centraremos en las *snakes* para explicar los modelos deformables paramétricos porque su uso está ampliamente extendido y el funcionamiento básico de otros modelos más complejos (como los de las superficies) es similar.

Las *snakes* o contornos activos son curvas definidas dentro del dominio de una imagen que pueden moverse por la influencia de las fuerzas internas y externas. Se espera que la combinación de los dos tipos de fuerzas lleve a la curva a ajustarse al contorno del objeto a segmentar. Los contornos activos son ampliamente usados en multitud de aplicaciones como segmentación de imágenes, modelado de formas y seguimiento de objetos en movimiento.

Si bien la segmentación de imágenes mediante contornos activos suele dar buenos resultados en muchas aplicaciones, también presenta algunas dificultades típicas. En concreto, los algoritmos de contornos activos sufren dos problemas fundamentales. En primer lugar, la posición inicial de la curva debe encontrarse relativamente cerca del borde del objeto a segmentar. En caso contrario, lo más probable es que el contorno no converja hacia el lugar deseado. El segundo problema es que la mayoría de los contornos activos tienen problemas para converger alrededor de objetos con bordes de forma cóncava. Este problema se debe a que las fuerzas externas no logran vencer a las fuerzas internas que tratan de mantener la forma de la curva lo más suave posible. Se puede aumentar el peso de las fuerzas externas (fuerzas de presión) para obligar a que la curva se ajuste a esas zonas cóncavas pero en muchos casos ocurre que los bordes más débiles no consiguen sujetar a la curva debido al empuje excesivo de estás fuerzas adicionales.

Como hemos comentado antes, una *snake* es un modelo deformable paramétrico que se mueve por el dominio espacial de una imagen. Definimos la curva como

$$\mathbf{x}(s) = [x(s), y(s)] \ , \tag{2.30}$$

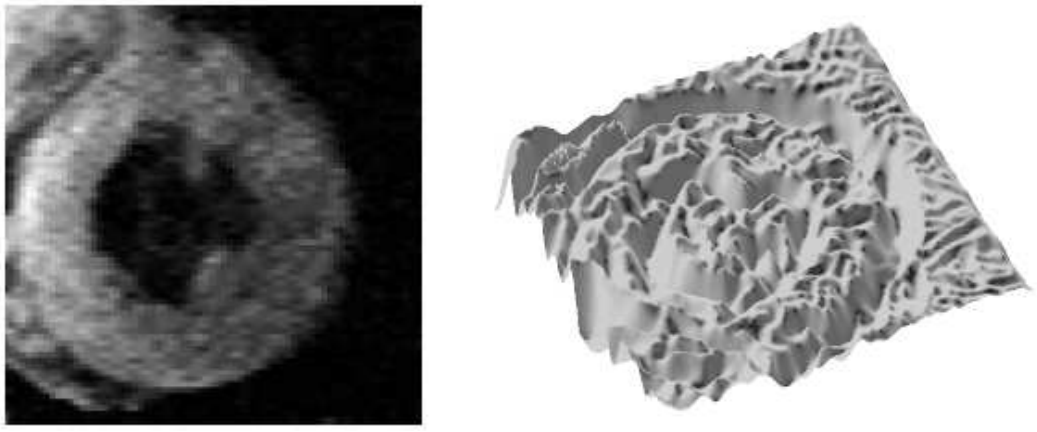
donde $s \in [0, 1]$.

El movimiento de una de estas curvas es tal que siempre trata de minimizar una función objetivo. Esta función objetivo es el funcional de energía definido por

$$E = \int_0^1 \frac{1}{2}(\alpha\left|\mathbf{x}'(s)\right|^2 + \beta\left|\mathbf{x}''(s)\right|^2) + E_{ext}(\mathbf{x}(s))\ ds \ , \tag{2.31}$$

donde α y β son parámetros que controlan la tensión y rigidez de la curva respectivamente. $\mathbf{x}(s)'$ y $\mathbf{x}(s)''$ son la primera y segunda derivadas de $\mathbf{x}(s)$ respecto a s. La función de energía externa E_{ext} se define a partir de la información de la imagen de forma que va tomando valores más pequeños

conforme la curva se aproxima a las zonas de interés. En la figura 2.38(b) podemos ver la representación de la función de energía sobre el dominio de la imagen 2.38(a). Puede observarse que la superficie de la función presenta mínimos en los bordes de interés de la imagen que arrastrarán a la curva hacia ellos.



(a) (b)

Figura 2.38.: Funcional de energía de una imagen. (a) Resonancia magnética de un ventrículo humano (b) Funcional de energía obtenido a partir de (a). La superficie presenta mínimos en los bordes de interés de la imagen. Imágenes tomadas de [14]

Llamando $I(x, y)$ a una imagen de niveles de grises (considerando valores continuos de x e y) las formas típicas de definir las energías externas para que la curva converja a los bordes de la imagen son

$$E^1_{ext}(x, y) = -\left|\nabla I(x, y)\right|^2 \ , \tag{2.32}$$

$$E^2_{ext}(x, y) = -\left|\nabla (G_\sigma(x, y) * I(x, y))\right|^2 \ , \tag{2.33}$$

donde $G_\sigma(x, y)$ es una gaussiana en dos dimensiones de desviación estándar σ y ∇ es el operador gradiente (ver sección 2.3.2, operador *LoG*). De las ecuaciones anteriores se deduce que, para valores altos de σ, los bordes se difuminan y distorsionan. Sin embargo, en ocasiones es necesario usar Gaussianas muy anchas para lograr que la curva "sienta" la proximidad de los bordes a una distancia considerable. De esta forma se puede aumentar el rango de captura de la *snake*.

Para que un contorno activo minimize la función E debe cumplir la ecuación de Euler:

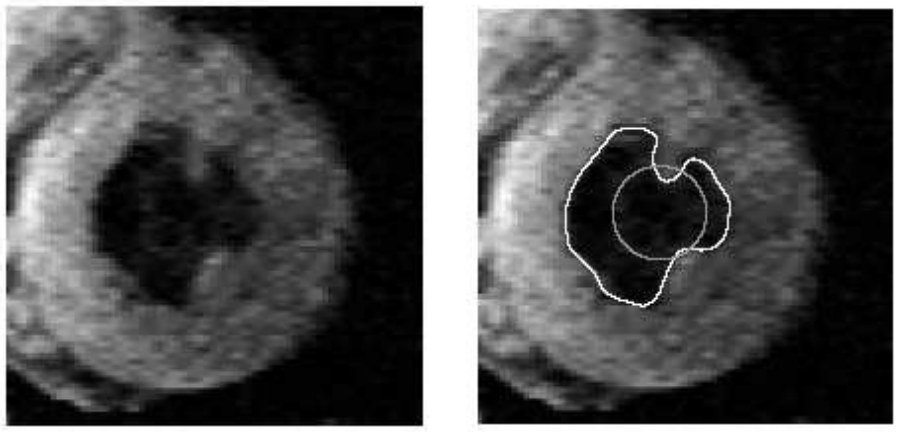
$$\alpha \mathbf{x}''(s) - \beta \mathbf{x}''''(s) - \nabla E_{ext} = 0 \ . \tag{2.34}$$

Esto puede interpretarse como un balance de fuerzas de la forma:

$$F_{int} + F^1_{ext} = 0 \ , \tag{2.35}$$

donde $F_{int} = \alpha \mathbf{x}''(s) - \beta \mathbf{x}''''(s)$ y $F_{ext}^1 = -\nabla E_{ext}$. Las fuerzas internas F_{int} tratan de mantener a la curva como está. Es decir, intentan evitar que la curva se estire o se doble. Las fuerzas externas F_{ext}^1 por el contrario, tratan de arrastrar a la curva hacia el borde del objeto de interés. Para satisfacer la ecuación 2.34 se dota a la curva $\mathbf{x}(s)$ de dinámica haciéndola depender del tiempo ($\mathbf{x}(s,t)$). Derivando respecto al tiempo y mediante un algoritmo de descenso por gradiente se puede, finalmente, resolver la ecuación 2.31.

Una variante de las *snakes* es la comúnmente conocida como *balloons*. Hemos comentado anteriormente que, en ocasiones es conveniente añadir a las fuerzas externas (dependientes de la información de la imagen) unas fuerzas de presión para conseguir que la curva converja hacia los bordes de interés. La alternativa de los *balloons* usa estas fuerzas de presión, siempre normales al contorno activo, con la intención de converger lo más rápidamente hacia los bordes de interés, reduciéndo así su dependencia de las condiciones iniciales. En estas curvas, el *balloon* (globo, en inglés) trata de maximizar su área al mismo tiempo que suaviza su contorno y busca las zonas de la imagen con mayor intensidad de gradiente. Se comporta, por lo tanto, como un globo hinchándose (o contrayéndose, si así se desea) hasta llegar a cubrir los bordes del objeto de interés.



(a) (b)

Figura 2.39.: Contorno activo balloon. (a) Resonancia magnética de un ventrículo. Nótese el bajo grado de resolución de la imagen. (b) El contorno activo inicialmente circular se expande hasta encontrar los bordes de interés. La precisión de la segmentación es claramente superior al nivel de pixelado de la imagen.

En la figura 2.39(a) podemos ver la resonancia magnética del ventrículo izquierdo de un corazón humano. El comportamiento de un contorno activo del tipo *balloon* ajustándose a los bordes significativos de la imagen se ilustra en la figura 2.39(b). El círculo es la forma inicial del contorno. Cuando el algoritmo comienza, el contorno activo se hincha progresivamente hasta alcanzar los bordes de interés (aquellos puntos en la imagen con mayor intensidad de gradiente). Esta imagen también sirve para ilustrar la capacidad de este tipo de algoritmos para generar segmentaciones de precisión superior a la resolución de la imagen. Esta característica es particularmente interesante en la segmentación de imágenes obtenidas con resonancias magnéticas.

Al comienzo de la sección hemos comentado los dos problemas principales de la segmentación mediante contornos activos. La dependencia de la posición inicial y la dificultad del contorno para converger hacia los bordes cóncavos. Ahora que conocemos las fuerzas que rigen el movimiento de los contornos activos podemos ilustrar mejor estos problemas mediante un ejemplo y, posteriormente, exponer algunas de las soluciones propuestas.

Hemos visto que las fuerzas externas son las que llevan a la curva a los bordes de interés. Estas fuerzas (en ausencia de fuerzas de presión adicionales) dependen exclusivamente de la naturaleza de la imagen y del funcional de energía derivado de ella. Si representamos este funcional en forma de campo de fuerza (complementando así la figura 2.38) distribuido a lo largo del dominio espacial de la imagen podremos comprender mejor el motivo de los problemas de esta técnica. En la figura 2.40(a) se muestra una figura en forma de U. El problema está en la cavidad de la forma, como vemos en la figura 2.40(b). La *snake* es incapaz de penetrar en la zona cóncava. Si echamos un vistazo a la representación de las fuerzas externas (figuras c y d) podemos comprender por qué. En la ampliación de las líneas de campo de la zona cóncava (figura (d)) vemos que las fuerzas tienen direcciones opuestas de manera que la curva nunca será arrastrada hacia adentro. Por otra parte, vemos que las líneas de fuerza tienen un rango de acción bastante limitado a las cercanías de los bordes de la imagen.

Es fácil imaginar el problema que supondría un contorno activo inicializado lejos de estas zonas. Para solucionar estos problemas se han propuesto diversos tipos de fuerzas externas. En la siguiente sección se comentan algunas de ellas.

2.6.2. Fuerzas externas

En las secciones anteriores se han expuesto los principios de funcionamiento de los modelos deformables y también sus principales dificultades. Las propuestas más conocidas para la solución de los problemas de convergencia de los modelos deformables se basan en la creación de nuevos tipos de fuerzas externas. A continuación vemos algunas de ellas:

- Fuerzas de Gaussiana con multiescala: Hemos comentado antes la influencia de la anchura de la Gaussiana σ en el rango de actuación de los contornos activos. Sin embargo el valor de σ no puede ser aumentado en exceso porque provocaría la distorsión de los bordes. La propuesta de la Gaussiana con multiescala consiste en crear un campo de fuerzas externas inicial usando un valor grande para σ. Cuando la curva converge, el campo de fuerzas se vuelve a calcular para una σ menor, y así sucesivamente. Mediante este procedimiento se aumenta considerablemente el rango de acción de los contornos. El problema de esta estrategia es que no proporciona un método claro para secuenciar los cambios de escala y, por lo

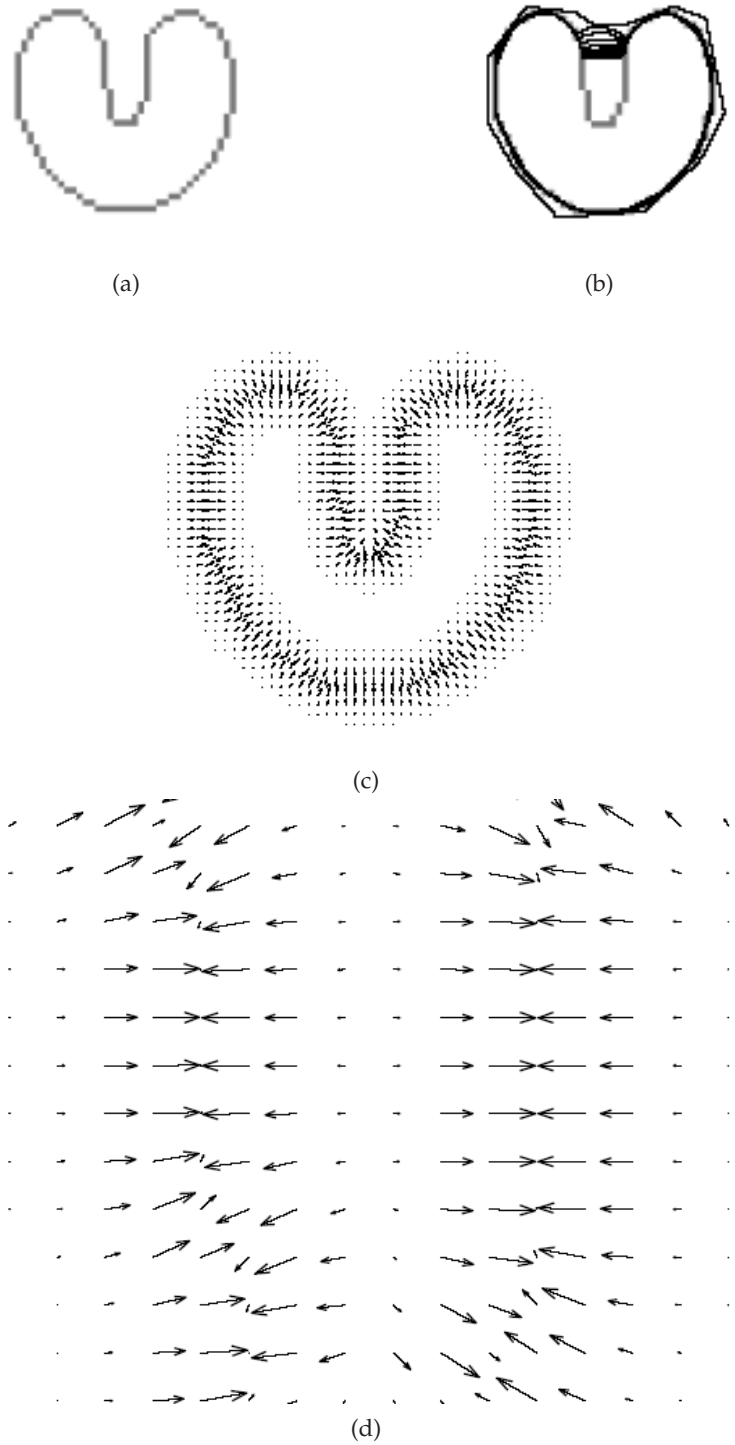
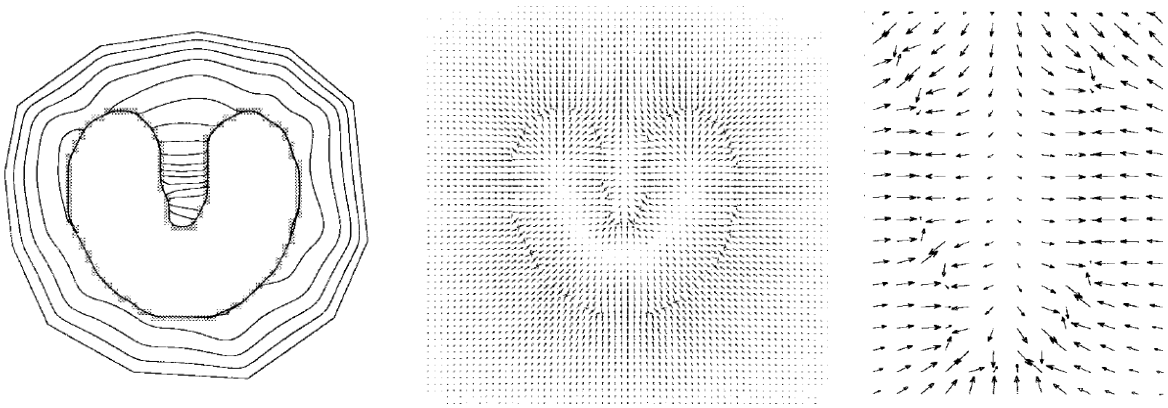


Figura 2.40.: Problema de convergencia de los contornos activos sobre bordes cóncavos. (a) Imagen. (b) La curva es incapaz de rodear la zona cóncava. (c) Representación del campo de fuerzas externas de la imagen. (d) Ampliación de la concavidad. Imágenes tomadas de [14]

tanto, precisa de un diseño *ad hoc* para cada problema.

- Fuerzas de presión: La fuerzas de presión empujan al contorno activo siempre en dirección normal a él. De esta forma las fuerzas de presión pueden inflar o contraer el contorno consiguiendo así que éste llegue a dar con los bordes de interés aunque se encuentren lejos inicialmente. El sentido de las fuerzas de presión (expandir o contraer el contorno) suele ser elegido por el usuario dependiendo de si el objetivo se encuentra dentro del contorno o fuera. El contorno sigue creciendo (o contrayéndose) hasta que es detenido por las fuerzas externas derivadas de la propia imagen (ver figura 2.39). En ocasiones, los contornos que usan este tipo de fuerzas (*balloons*) corren riesgo de cruzarse formando rizos.

- GVF (*Gradient Vector Flow*): Esta fuerza externa usa unas ecuaciones de difusión sobre el gradiente de la imagen que modifican el mapa de fuerzas alargando las líneas de campo de forma proporcional a la fuerza del borde. Este nuevo mapa se conoce como campo GVF. Se demuestra que esta técnica mejora la convergencia de los contornos sobre bordes cóncavos y aumenta el rango de acción de éstos. En la figura 2.41 podemos ver un ejemplo donde el contorno activo logra converger hacia el interior de la cavidad. Como muestran las figuras 2.41(b) y (c), el campo de fuerzas generado mediante este método tiene un mayor rango de acción y las líneas de campo generadas en la concavidad sí tiran de la *snake* hacia abajo. Si en lugar de representar el campo creado, comparamos las líneas de flujo se hacen todavía más evidentes las ventajas que ofrecen las fuerzas GVF. Las líneas de flujo representan la dirección que seguiría una partícula libre situada en un punto del campo de fuerzas. En la figura 2.42 hemos representado las líneas de flujo creadas por un campo con fuerzas externas convencionales frente a las líneas de flujo creadas por un campo de fuerzas usando GVF. Las diferencias son evidentes.

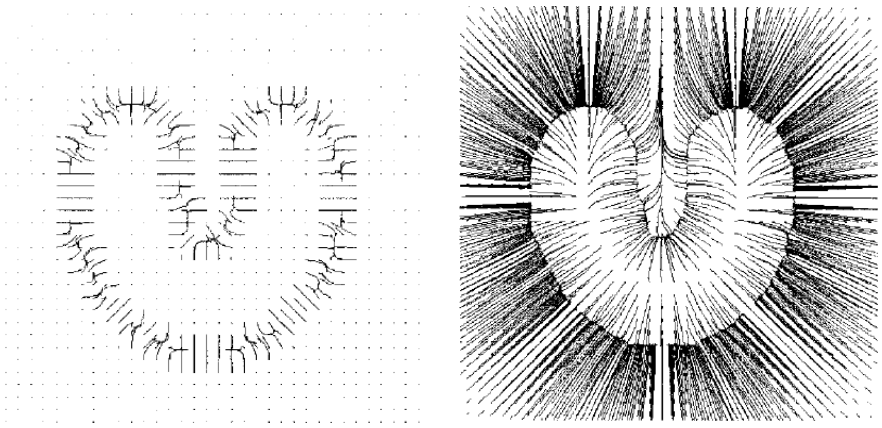


(a) (b) (c)

Figura 2.41.: (a) Convergencia de un contorno activo usando la fuerza externa GVF. (b) Campo GVF. (c) Ampliación de la concavidad. Imágenes tomadas de [14]

Además de las fuerzas externas citadas anteriormente existen otras como las *fuerzas basadas en mapas de distancia*, cuyo campo de fuerzas es función de la distancia a los bordes de cada píxel

de la imagen. En algunos casos es deseable que el usuario sea capaz de interactuar con el contorno activo, influyendo en su trayectoria. Estas *fuerzas interactivas* actúan sobre la curva atrayéndola o empujándola hacia o desde unos puntos clave definidos por el usuario.



(a) (b)

Figura 2.42.: (a) Líneas de flujo creadas por un campo de fuerzas convencional. (b) Líneas de flujo creadas por un campo de fuerzas GVF. Imágenes tomadas de [14]

En la figura 2.43 podemos ver una comparación entre los comportamientos de un contorno sobre el que actúan diferentes tipos de fuerzas externas. La figura 2.43(b) muestra la convergencia de un contorno influenciado por fuerzas de presión. Como vemos, la curva tiende a hincharse hasta ajustarse a los bordes (incluso en las zonas cóncavas) de la imagen pero se desborda en las zonas donde no hay borde resultando un contorno que no se ajusta a la forma real que representa la imagen. La curva de la figura 2.43(c) se mueve influenciada por fuerzas externas basadas en mapas de distancia. Esta curva no consigue llegar a las concavidades de la figura porque el fondo de éstas esta demasiado lejos y las fuerzas internas junto con la atracción de los bordes más cercanos sujetan al contorno impidiéndole llegar hasta los bordes izquierdo y derecho. La fuerza externa que mejor responde a la imagen de este ejemplo es la que vemos en la figura 2.43(d). Las fuerzas GVF difunden las líneas de flujo de forma que la curva siempre es atraída hasta el fondo de las concavidades.

Comentaremos finalmente que los modelos paramétricos deformables, a pesar de haber tenido éxito en un amplio rango de aplicaciones, tienen dos limitaciones principales. En situaciones donde el modelo inicial difiere mucho en forma y tamaño respecto al objeto de interés, los parámetros deben ser recalculados dinámicamente para segmentar el objeto. Los métodos para recalcular parámetros en modelos de dos dimensiones no suelen ser demasiado complicados y el tiempo de proceso que requieren es moderado. Sin embargo, para modelos en tres dimensiones (superficies o mallas activas) este proceso suele ser complejo y computacionalmente caro. Otra limitación importante de los modelos paramétricos deformables ya ha sido comentada anteriormente. Las adaptaciones topológicas como la división o fusión de los contornos es complicada ya que requiere la construcción de un nuevo modelo paramétrico.

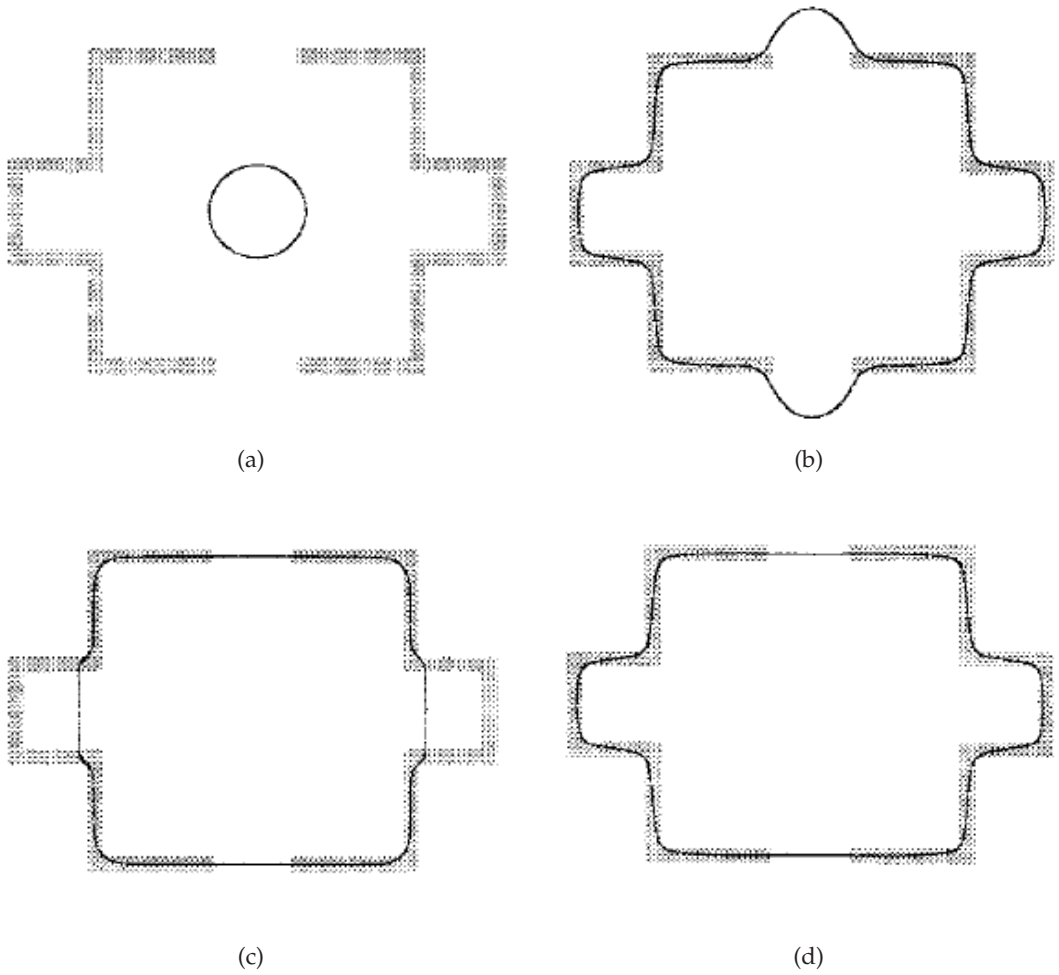


Figura 2.43.: Comparación de resultados para diferentes tipos de fuerzas externas. (a) Situación inicial de la curva (b) Convergencia final usando fuerzas de presión. (c) Convergencia final usando fuerzas basadas en mapas de distancia (d) Convergencia final usando las fuerzas GVF. Imágenes tomadas de [14].

2.7. Segmentación de imágenes en movimiento

El movimiento es una fuente de información valiosísima para los sistemas de visión en seres humanos y animales. Todos sabemos que es mucho más fácil encontrar un objeto en movimiento que uno que permanece estático, especialmente cuando el objeto de interés es parecido al fondo sobre el que se encuentra. En los sistemas de visión artificial que requieren la segmentación de imágenes en movimiento, éste se entiende como el desplazamiento relativo entre el sistema de captación del movimiento (el objetivo de la cámara) y la escena que está captando.

En los últimos años se ha intensificado la investigación de las técnicas de segmentación de video. Las nuevas aplicaciones de comunicaciones multimedia como la videoconferencia han cambiado la forma de representar las imágenes de vídeo. La necesidad de transmitir vídeo consumiendo el mínimo ancho de banda posible ha provocado un cambio radical en el enfoque de la codificación de vídeo. Los nuevos estándares de ISO, MPEG-4 y MPEG-7 proponen una codificación basada en el contenido de la imagen en lugar del tradicional enfoque basado exclusivamente en los fotogramas (*frames*). MPEG-4 define una sintaxis para una serie de funciones basadas en el contenido de los videos, entre ellas, la interactividad del usuario con las imágenes. Estas nuevas funciones dejan obsoleta la codificación de las imágenes como simples matrices de píxeles y orienta la investigación hacia la creación de objetos, es decir, entidades con significado semántico. No obstante, no especifica como ha de generarse la codificación de los datos de vídeo. Para generar estos datos, las imágenes han de segmentarse en objetos. Sobre estos objetos se realizará un seguimiento (*tracking*) de su movimiento relativo de un fotograma a otro.

Esta y otras aplicaciones han motivado el estudio y revisión de las técnicas de segmentación de imágenes y vídeo. En la siguiente sección veremos los principios básicos de la segmentación de vídeo.

2.7.1. Técnicas basadas en el dominio del espacio

Una de las aproximaciones más simples para la detección de cambios entre dos fotogramas $f(x, y, t_i)$ y $f(x, y, t_j)$ tomados en diferentes instantes de tiempo (t_i y t_j respectivamente), es la comparación de las dos imágenes píxel a píxel. Un procedimiento bastante común para realizar esta comparación es calcular la *imagen diferencia* de los dos fotogramas. Supongamos que tenemos una imagen referencia que únicamente contiene componentes estacionarios. La imagen diferencia resultante de la comparación de este fotograma frente a otro igual, pero con un objeto en movimiento será una imagen cuyos píxeles serán todos cero excepto los correspondientes al objeto no estático.

La imagen diferencia resultante de otros dos fotogramas tomados en instantes t_i y t_j puede definirse como

$$d_{ij}(x,y) = \begin{cases} 1 & \text{si } |f(x,y,t_i) - f(x,y,t_j)| > \theta \\ 0 & \text{en cualquier otro caso} \end{cases} , \tag{2.36}$$

donde θ es un umbral. Nótese que $d_{ij}(x,y)$ será en las coordenadas espaciales (x,y) sólo si la diferencia entre los niveles de grises entre las dos imágenes en ese punto es apreciable (mayor que θ). Aquellos píxeles en $d_{ij}(x,y)$ con valores 1 serán considerados como el resultado del movimiento del objeto. Esta técnica sólo es aplicable si disponemos de las dos imágenes y la iluminación es relativamente constante dentro de los márgenes establecidos por θ. En la práctica, muchos de los píxeles con valores igual a 1 en $d_{ij}(x,y)$ son provocados por el ruido. Normalmente estos píxeles son puntos aislados en la imagen diferencia y pueden ser eliminados fácilmente formando regiones que sólo contengan 1's con conectividad de 4 u 8 vecinos. Aquellos píxeles que no cumplan dicha condición pueden ser eliminados. Mediante esta práctica podemos perder el movimiento de objetos pequeños o lentos pero suele dar buenos resultados en la mayor parte de los casos.

La figura 2.44 ilustra estos conceptos. En la figura 2.44(a) vemos una imagen referencia tomada en el instante t_i. En la figura 2.44(b) vemos una imagen tomada en el instante t_j y la figura 2.44(c) es la imagen diferencia. Todos los píxeles de la imagen resultante están a cero excepto aquellos que corresponden al vehículo en movimiento. Además puede observarse el efecto del ruido, que introduce píxeles no nulos distribuidos aleatoriamente por el dominio de la imagen. Estos píxeles espurios pueden ser eliminados mediante técnicas basadas en conectividad como hemos comentado anteriormente.

(a) (b) (c)

Figura 2.44.: (a) Imagen de referencia tomada en el instante t_i. (b) Imagen tomada en el instante t_j. (c) Imagen diferencia.

Además de las técnicas para segmentación de video basadas en el espacio existen técnicas que se basan en el dominio de la frecuencia. Estos métodos usan transformadas de Fourier sobre la imagen y codifican las variaciones entre los fotogramas según esta formulación. No obstante, la mayoría de las aproximaciones propuestas se basan en la segmentación espacio temporal.

En [21], podemos encontrar ambiciosas e interesantes propuestas para segmentación de vídeo orientado a aplicaciones multimedia. Se pretende mediante sofisticadas técnicas, no sólo que la segmentación y seguimiento sea coherente con las características y descriptores estadísticos de los fotogramas, sino que la segmentación se corresponda a objetos con significado semántico. Estos algoritmos, de forma supervisada o no supervisada, filtran la imagen para simplificar la información y tratar sólo con los datos relevantes a la hora de decidir la segmentación. Se combinan diversas técnicas para distinguir entre las regiones en movimiento y estáticas, etiquetándolas y asignándoles una entidad dentro de la imagen. Este etiquetado sirve para la estimación y el seguimiento de estos objetos. Esta gestión de la segmentación espacial a lo largo del tiempo se lleva a cabo mediante árboles de partición, similares a los *quadtrees* vistos en la sección 2.5.2. La figura 2.45 puede dar una idea intuitiva de la filosofía de estas técnicas, aunque el estudio a fondo de éstas es bastante más complicado.

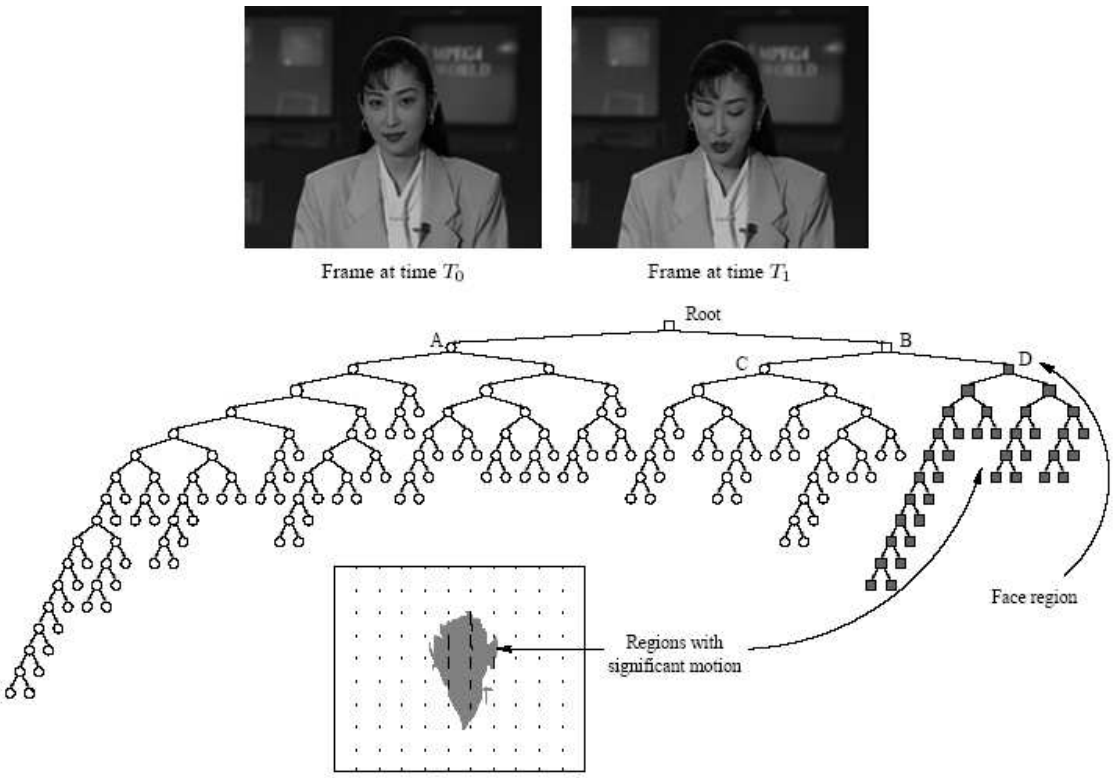


Figura 2.45.: Estimación/segmentación de movimiento mediante el Árbol de partición. De [21]

En la parte superior de la figura 2.45 se muestran dos fotogramas tomados en instantes de tiempo consecutivos T_0 y T_1. El nodo D está asignado a la cara y mientras que los otros nodos están asignados a otras partes significativas de la imagen. El movimiento detectado corresponde al nodo D y sus subregiones, de forma que el sistema puede estimar/segmentar el movimiento de la cara de la presentadora.

2.8. Conclusiones. Estado del arte.

La segmentación de imágenes es el paso preliminar más importante en la mayoría de los sistemas de visión artificial. Como hemos visto en todos los ejemplos indicados en las secciones anteriores, la elección de una técnica u otra depende en gran medida de las características del problema. Las técnicas revisadas en las secciones anteriores son las más representativas de la amplia variedad de aproximaciones que pueden encontrarse al problema de la segmentación de imágenes.

El estado del arte en el campo de la segmentación de imágenes está representado, en la mayoría de los casos, por algoritmos que son una combinación de varias técnicas menos sofisticadas. En las siguientes secciones nos centraremos en el algoritmo de S.C. Zhu y A. Yuille llamado *Region Competition* [1] y su desarrollo y adaptación para la segmentación de imágenes estereoscópicas de vídeo. Este algoritmo combina características de los contornos activos y las técnicas basadas en regiones. En los últimos años, este artículo ha servido, a su vez, de base para investigaciones posteriores de los mismos autores y otros muchos.

Zhu y Yuille presentaron en [24] un interesante método para reconocimiento de objetos flexibles animados a partir de unas formas básicas predefinidas. Una optimización del algoritmo RC para la segmentación de texturas basándose en histogramas de patrones binarios locales puede encontrarse en [20]. También el propio Zhu, junto a otros autores presentó un estudio para el modelado de texturas basándose en campos aleatorios de Markov (MRF) en [23]. Los modelos MRF son particularmente adecuados para el modelado de texturas ya que éstas se caracterizan por ser repeticiones de globales (pero no predecibles) de un patrón local. En [23] se combinan estos modelos con técnicas de filtrado y máxima entropía. En [19] se propone un algoritmo GSRC (*General Scheme of Region Competition Based on Scale Space*) en el que se combinan las atractivas características de RC con un nuevo algoritmo de clasificación basado en el *clustering* del espacio de parámetros de la imagen a diferentes escalas.

En cuanto a la segmentación de imágenes de video, las investigaciones se han desarrollado mucho en los últimos años debido a la creación de nuevos estándares de video. Puede encontrarse un interesante estudio de la situación en este campo en [21]. Más recientemente, en [22] encontramos una nueva aproximación para el seguimiento de objetos en vídeo basada en contornos activos.

Las referencias anteriores son sólo la punta del iceberg de lo que puede encontrarse en la literatura reciente. Diversos estudios se suceden año tras año abordando el problema de la segmentación y reconocimiento de objetos desde diferentes perspectivas.

Capítulo 3.

Competición de Regiones para la segmentación de músculos artificiales

3.1. Competición de Regiones de Zhu y Yuille

En la primera parte de esta memoria, hemos visto diferentes aproximaciones para la solución del problema de segmentación de imágenes. Técnicas basadas en la detección de bordes, umbrales, crecimiento de regiones, contornos activos, todas ellas tienen algo en común: formulan hipótesis acerca de la imagen, evalúan alguna característica de ésta y toman decisiones aplicando algún tipo de umbral explícita o implícitamente. La principal diferencia entre las diferentes estrategias es el dominio sobre el que se basan estas hipótesis y decisiones. Cada una de estas estrategias tiene sus desventajas. Las técnicas basadas en detección de bordes (ver sección 2.3) solamente usa información local y no puede garantizar bordes continuos y totalmente cerrados sin ningún postprocesado adicional. Los modelos de contornos activos como *snakes* y *balloons* (ver sección 2.6) sólo usan la información cercana al propio contorno y además necesitan ser inicializadas en una buena posición respecto al borde que se quiere detectar. El crecimiento de regiones (ver sección 2.5) tiene la ventaja de que evalúa las estadísticas dentro de cada región en cada iteración pero a menudo produce bordes irregulares y pequeños agujeros. Los métodos que tratan de usar un criterio global (Bayes/MDL) (ver el anexo A) tienen a menudo problemas para encontrar los mínimos de la función de energía.

Song Chun Zhu y Alan Yuille [1] presentaron en 1996 un nuevo entorno de trabajo para la segmentación de imágenes al crear un algoritmo llamado *Competición de Regiones*. Este algoritmo combina las atractivas características geométricas de los contornos activos con las técnicas estadísticas del crecimiento de regiones. Usa además una ventana de muestreo cuyo tamaño depende del nivel de ruido presente en la imagen y determinará la precisión de los bordes resultantes en la segmentación final. Al igual que ocurre con muchos otros algoritmos (por ejemplo, Canny [11], ver sección 2.3.4) si usamos ventanas de muestreo grandes se mejorará el comportamiento frente

al ruido pero los bordes resultantes serán mucho menos precisos.

Como en otros muchos algoritmos, el buen rendimiento de la competición de regiones depende de las condiciones iniciales. Estas condiciones iniciales son, más concretamente, la posición de las semillas iniciales. Zhu y Yuille propusieron un método por el cual las malas semillas acaban transformándose en buenas semillas.

El algoritmo puede generalizarse directamente para segmentación multibanda (para imágenes en color) y también para el uso de diferentes descriptores como la textura de los materiales. Aunque nos centraremos en la segmentación basada en los niveles de grises, esta capacidad del algoritmo es relevante ya que le otorga la capacidad de segmentar imágenes con discontinuidades en los niveles de grises causadas por sombras y diferentes tipos de iluminación.

3.1.1. La competición de regiones

El objetivo de la segmentación de imágenes es realizar particiones que separen las regiones con características homogéneas de intensidad. Se espera que estas regiones se correspondan con objetos o partes de los objetos. Antes de comenzar con la descripción del algoritmo necesitamos definir claramente qué es una región homogénea.

Diremos que una región R es homogénea cuando sus valores de intensidad sean susceptibles de ser generados mediante una o un grupo de distribuciones de probabilidad $P(I|\alpha)$ predefinidas, donde α son los parámetros de la distribución. Se asume que los modelos probabilísticos pueden diferir de unas regiones a otras. El uso de diferentes modelos es útil en la segmentación de imágenes con zonas muy diferentes y que, por lo tanto, han de ser descritas por diferentes distribuciones.

Supongamos ahora que todo el dominio de la imagen R ha sido segmentado inicialmente en M subregiones R_i homogéneas donde $i = 1, 2, \ldots, M$ y se cumple que $R = \cup_{i=1}^{M} R_i$, $R_i \cap R_j = \emptyset$ si $i \neq j$.

Sea ∂R_i el borde de una región R_i cuya dirección se define en sentido opuesto a las agujas del reloj, de forma que si viajamos a lo largo del borde Γ_i, R_i queda a nuestra izquierda. Definimos también $\Gamma = \cup_{i=1}^{M} \Gamma_i$ como los bordes de la segmentación de toda la imagen, donde $\Gamma_i = \partial R_i$.

Consideramos ahora un criterio MDL (funcional de energía global) de la forma

$$E\left[\Gamma, \{\alpha_i\}\right] = \sum_{i=1}^{M} \{ \frac{\mu}{2} \oint_{\partial R_i} ds - \log P\left(\left\{I_{(x,y)} : (x,y) \in R_i\right\} | \alpha_i\right) + \lambda \}\ , \tag{3.1}$$

donde, en el segundo miembro, el primer término dentro de las llaves representa la longitud de la curva ∂R_i para la región R_i. Asumimos que la longitud de código es proporcional a la longitud de la curva y μ es la longitud de código para un arco de la curva de longitud unitaria. Como todos los bordes están compartidos por dos regiones adyacentes, se divide este primer término entre dos. El segundo término representa la suma del coste en código para la intensidad de cada píxel (x, y) dentro de la región R_i de acuerdo con la distribución $P\left(\left\{I_{(x,y)} : (x, y) \in R_i\right\} |\alpha_i\right)$. Por último, λ es la longitud de código necesaria para describir la distribución de cada región R_i y, simplemente, se considera de igual valor para todas las regiones.

Como vemos, la ecuación 3.1 depende de dos grupos de variables: la segmentación Γ y los parámetros α_i. Esta circunstancia sugiere el uso de un algoritmo que trabaje en dos etapas alternas. La primera etapa minimiza la energía localmente en dos pasos, manteniendo constante el número de regiones. La segunda etapa funde regiones siempre que la fusión consiga reducir la energía.

La primera etapa consiste en lo siguiente:

En el primer paso, fijamos Γ, es decir, fijamos R_i y $I_(x, y)$, y buscamos los descriptores α_i que minimizan el coste de código para cada región. Según la regla de Bayes, esto es

$$\alpha_i^* = \arg\min_{\alpha_i}\{-\int\int_{R_i} log\ P(\alpha_i|I_(x, y))dxdy\},\ \forall i\ . \tag{3.2}$$

Para el caso discreto

$$\alpha_i^* = \arg\max_{\alpha_i} \prod_{(x,y)\in R_i} P(\alpha_i|I_(x, y)),\ \forall i\ . \tag{3.3}$$

En otras palabras, se estiman los α_i maximizando las probabilidades condicionales. Para muchas distribuciones esto puede hacerse analíticamente. Por ejemplo, si $P(I|\alpha_i)$ es la distribución Gaussiana el cálculo de los α_i^* es simplemente la media y varianza de los píxeles dentro de R_i.

En el segundo paso se fijan los descriptores α_i y se realiza el descenso por gradiente respecto a Γ. Para cualquier punto $\vec{v} = (x, y)$ del borde Γ se obtiene:

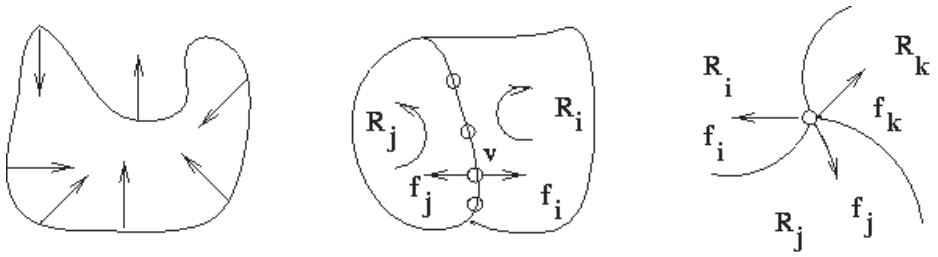
$$\frac{d\vec{v}}{dt} = -\frac{\partial E[\Gamma_i, \{\alpha_i\}]}{\partial \vec{v}}\ , \tag{3.4}$$

donde el segundo miembro es la derivada del funcional de energía de la ecuación 3.1.

Realizando esta derivada se obtiene la ecuación de movimiento del punto $\vec{v}$

$$\frac{d\vec{v}}{dt} = \sum_{k\in Q_{(\vec{v})}} \{-\frac{\mu}{2}\kappa_{k(\vec{v})}\vec{\mathbf{n}}_{k(\vec{v})} + \log P(I_{(\vec{v})}|\alpha_k)\vec{\mathbf{n}}_{k(\vec{v})}\}\ , \tag{3.5}$$

donde $Q_{(\vec{v})} = \{k \mid \vec{v} \text{ está sobre } \Gamma_k\}$, el sumatorio se calcula sobre aquellas regiones R_k para las que $\vec{v}$ está sobre Γ_k. $\kappa_{k(\vec{v})}$ es la curvatura de Γ_k en el punto $\vec{v}$ y $\vec{\mathbf{n}}_{k(\vec{v})}$ es el vector unitario con dirección normal a Γ_k en el punto $\vec{v}$. Asumimos que, según el sentido en el que recorremos los bordes (al contrario de las agujas del reloj), el vector unitario $\vec{\mathbf{n}}_k$ apunta hacia afuera de R_k.



(a) (b) (c)

Figura 3.1.: Fuerzas sobre el contorno. (a) Fuerzas de suavizado. (b) Fuerzas estadísticas en un punto del borde. (c) Fuerzas estadísticas en un punto de unión.

La ecuación 3.5 tiene una interpretación simple a la vez que intuitiva. Existen dos tipos de fuerzas actuando sobre el contorno, ambas tirando de él en la dirección normal. El primer término, la fuerza de *suavizado*, es más fuerte en los puntos que presentan mayor curvatura. La figura 3.1(a) muestra la acción de las fuerzas de suavizado sobre los puntos del borde de una región. Esta fuerza es independiente de la dirección de la curva y siempre trata de mantenerla lo más recta posible. El segundo término representa las fuerzas *estadísticas*, $\vec{f} = \log P(I|\alpha)\vec{\mathbf{n}}$. Como el logaritmo de P siempre será menor o igual que cero, las fuerzas estadísticas siempre tratan de comprimir la región. Cuanto mejor satisfaga el punto $\vec{v}$ la condición de homogeneidad mayor será $P(I|\alpha)$ y más débiles serán las fuerzas estadísticas.

Por ejemplo, en la figura 3.1(b), $\vec{v}$ es un punto en la frontera común de las regiones R_i y R_j. Como los contornos Γ_i y Γ_j tienen vectores normales de dirección opuesta en el punto $\vec{v}$, tenemos que $\vec{\mathbf{n}_i} = -\vec{\mathbf{n}_j}$ y $\kappa_i\vec{\mathbf{n}}_i = \kappa_j\vec{\mathbf{n}}_j$. La ecuación de movimiento para $\vec{v}$ es:

$$\begin{aligned}\frac{d\vec{v}}{dt} &= -\mu\kappa_{i(\vec{v})}\vec{\mathbf{n}}_{i(\vec{v})} + (\log P(I_{(\vec{v})}|\alpha_i) - \log P(I_{(\vec{v})}|\alpha_j))\vec{\mathbf{n}}_{i(\vec{v})} \\ &= -\mu\kappa_{i(\vec{v})}\vec{\mathbf{n}}_{i(\vec{v})} + \log\frac{P(I_{(\vec{v})}|\alpha_i)}{P(I_{(\vec{v})}|\alpha_j)}\vec{\mathbf{n}}_{i(\vec{v})}\ .\end{aligned} \tag{3.6}$$

En la ecuación 3.6 podemos ver de nuevo el término de suavizado y, además un segundo término determinado por un *test de verosimilitud*. Si $P(I_{(\vec{v})}|\alpha_i) > P(I_{(\vec{v})}|\alpha_j)$ (por ejemplo, si el nivel de intensidad en $\vec{v}$ se ajusta mejor a la distribución de la región R_i que a la de la región R_j) entonces el borde se moverá en la dirección de $\vec{\mathbf{n}}_i$. Ocurre lo mismo en aquellos puntos compartidos por bordes de varias regiones. En la figura 3.1(c) se muestra la acción de las fuerzas estadísticas en un

punto de unión entre tres regiones. Las regiones adyacentes compiten entre sí por los píxeles que se encuentran en su frontera. De ahí el nombre de *competición de regiones*.

Los dos pasos iniciales del algoritmo descritos anteriormente hacen que el funcional de energía disminuya. Además, la función está acotada inferiormente de forma que se garantiza la convergencia hacia un mínimo local. No obstante, esta primera etapa en dos pasos no permite alterar el número de regiones existentes. Por lo tanto, es necesario añadir una segunda etapa en la que dos regiones adyacentes se fundan siempre y cuando esto produzca una disminución de energía. Después se vuelve a la primera etapa de dos pasos y así sucesivamente. Finalmente, tras un número de iteraciones que siempre reducen la energía del funcional, se alcanza un mínimo local.

3.1.2. RC como combinación de varias técnicas

Hemos comentado anteriormente que el algoritmo *Region Competition* (RC) combina los mejores aspectos del crecimiento de regiones y los contornos activos (ver secciones 2.5 y 2.6) y, de hecho, muchas características de estos modelos pueden derivarse de casos especiales del algoritmo RC como vamos a ver a continuación.

En primer lugar, el crecimiento de regiones con umbral constante puede considerarse como un caso degenerado de la competición de regiones. En este supuesto, tratamos la región creciente como R_1 con una $P(I|\alpha_1)$ elegida de acuerdo con nuestro criterio de homogeneidad, y otra región de fondo como R_0 (cumpliéndose que $R_1 \cup R_0 = R$) con una distribución uniforme P_0. Tenemos, en esta situación, la siguiente ecuación de movimiento para cada punto $\vec{v}$:

$$\frac{d\vec{v}}{dt} = (\log P(I_{(\vec{v})}|\alpha_1) - \log P_0)\vec{\mathbf{n}}_{(\vec{v})} \ , \tag{3.7}$$

donde $\vec{\mathbf{n}}$ es el vector normal al contorno de la región. La aplicación de esta ecuación a una imagen digital se corresponde con el crecimiento de regiones donde la probabilidad $P(I_{(\vec{v})}|\alpha_1)$ se calcula y compara con el umbral absoluto P_0.

En segundo lugar, el supuesto estadístico inherente a los modelos de *snakes* y *balloons* es todavía más simple. En este caso tratamos la región objeto R_1 y el fondo R_0 como distribuciones uniformes P_1 y P_0, con $v = \log P_1 - \log P_0$ (donde v es la fuerza externa añadida que siempre trata de expandir el contorno en el caso de los *balloons*). Para el caso particular $v = 0$ se obtiene el modelo de *snakes*. Si en la ecuación 2.31 añadimos un término adicional de energía para modelar la fuerza externa que trata de expandir el contorno tenemos (la notación es ligeramente diferente):

$$E\left[\Gamma_{(s)}\right] = \oint_{\Gamma_{(s)}} \frac{1}{2}(\alpha\left|\Gamma_{(s)}\right|^2 + \beta\left|\Gamma_{(ss)}\right|^2) + \lambda\left|\vec{\nabla}I \cdot \vec{\nabla}I\right| \, ds - v\int\int_R dxdy \ , \tag{3.8}$$

que es el funcional de energía para los *balloons*. Derivando podemos obtener la ecuación de movimiento para un punto $\vec{v}$ del contorno:

$$\frac{d\Gamma_{(s)}}{dt} = -\alpha\Gamma_{(ss)} + \beta\Gamma_{(ssss)} + (\log P_1 - \log P_0)\vec{n_(s)} + \lambda\vec{\nabla}\left|\vec{\nabla}I \cdot \vec{\nabla}I\right| \ . \tag{3.9}$$

Esta ecuación difiere de la ecuación de movimiento de RC (ecuación 3.6), en dos aspectos. Las fuerzas de suavizado para los *balloons* (los dos primeros términos, ponderados por los coeficientes α y β) son ligeramente diferentes respecto a las de RC. Esta es una diferencia menor que puede ser fácilmente corregida. El término final en la ecuación 3.9 es un umbral para decidir cuándo debe dejar de moverse el contorno y está basado en medidas locales de los bordes presentes en la imagen. Este término no aparece en la ecuación de RC pero podría añadirse un término similar incluyendo un coste de codificación para los bordes de la imagen de la forma $-\oint_{\partial R} \log P(I_e|e)ds$ donde $P(I_e|e)$ es la probabilidad de medida de un borde (I_e), condicionada a que un borde e esté presente. En el caso de los *balloons*, su principal defecto es que no usa ninguna información estadística acerca de la región que rodea y esto provoca que, a menudo, fracase en sus intentos de segmentar una imagen en regiones homogéneas.

En resumen, el algoritmo RC combina los aspectos más atractivos de los contornos activos *snakes/balloons* y el crecimiento de regiones para minimizar una función de coste global, según un criterio MDL.

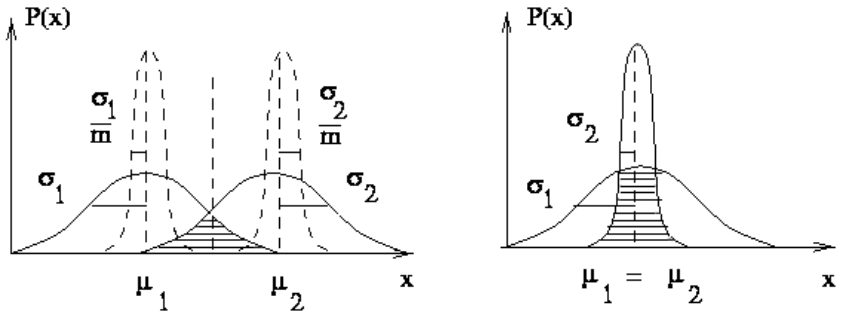
3.1.3. Generalizando el criterio MDL para RC

Para ilustrar el funcionamiento del algoritmo, consideraremos que $P(I|\alpha)$ es una distribución Gaussiana. Tenemos, por lo tanto, que $\alpha = (\mu, \sigma)$, y $P(I_{(x,y)}|(\mu,\sigma)) = \frac{1}{\sqrt{2\pi}\sigma}\exp\{-\frac{(I_{(x,y)}-\mu)^2}{2\sigma^2}\}$. De esta forma se simplifican las ecuaciones pero podrían usarse otras distribuciones más sofisticadas si así se desea.

Hemos visto en la sección anterior algunas de las características favorables que RC hereda de otras técnicas como *snakes* y el crecimiento de regiones. Sin embargo, si analizamos la ecuación 3.6 podemos encontrar dos principales desventajas que el algoritmo descrito hasta ahora comparte con estas otras técnicas.

En primer lugar, las fuerzas estadísticas en cada punto frontera (x, y) son función de $P(I_{(x,y)}|(\mu,\sigma))$, en otras palabras, la probabilidad de que $I_{(x,y)}$ se encuentre a un lado u otro del borde está determinada por la distribución Gaussiana $N(\mu, \sigma^2)$. Aunque esta definición de la fuerza estadística parece plausible no debemos olvidar que, al tomar sólo una muestra de esta distribución, existe una razonable posibilidad de que $I_{(x,y)}$ caiga en una de las colas de la distribución. Esta situación puede llevar a un error de clasificación del píxel en cuestión. Nuestras fuerzas

estadísticas serán entonces demasiado sensibles a las pequeñas variaciones en la intensidad. La figura 3.2(a) ilustra este caso. Las curvas continuas muestran dos distribuciones Gaussianas que se solapan, la zona rayada es la región donde puede darse el error de clasificación. Más adelante comentaremos la forma de soslayar este problema mediante el uso de ventanas.



(a) (b)

Figura 3.2.: Dos posibles distribuciones para regiones adyacentes. (a) Las líneas continuas representan dos distribuciones Gaussianas muy solapadas debido a las fluctuaciones del ruido, las líneas discontinuas son las distribuciones después de usar las ventanas de muestreo. (b) Dos Gaussianas con la misma media pero diferente varianza.

Otro problema surge en el caso extremo de que tengamos dos distribuciones con la misma media pero diferente varianza (ver figura 3.2). En estos casos es necesario medir los momentos de segundo orden (las varianzas) en la vecindad de cada punto frontera (x, y) para decidir a cuál de las dos distribuciones pertenece el punto. Nótese que si, usamos distribuciones más sofisticadas que la Gaussiana, el cálculo estadístico necesario en la vecindad del punto (x, y) será más complejo.

No obstante, estos problemas no se deben sólo a la naturaleza del algoritmo RC sino que son intrínsecos al criterio MDL. Por ejemplo, si los datos son generados por Gaussianas con una varianza alta, cualquier algoritmo tendrá dificultad para clasificar los píxeles a causa de las fluctuaciones de las muestras. Esto resultará en una segmentación errónea aunque se consiga alcanzar el mínimo global de MDL. Se demuestra (ver [18]) que si los datos de una región son generados por una Gaussiana de varianza σ^2 entonces la varianza del conjunto N de muestras generadas en la región es del orden de σ^4/N. Para regiones pequeñas, donde $N << \sigma^4$, resultará energéticamente favorable codificar esa región como dos o más conjuntos, independientemente del algoritmo que se use. Según esto, no se puede esperar que el criterio MDL estándar produzca soluciones correctas para regiones de este tamaño y tendrá siempre tendencia a sobresegmentar la imagen.

Zhu y Yuille propusieron una solución a estos problemas basada en el uso de ventanas de muestreo circulares de m píxeles de tamaño. Estos píxeles vecinos del punto (x, y) los denotaremos por $\mathcal{W}_{(x,y)}$. El propósito de estas ventanas es sustituir la distribución $P(I_{(x,y)}|\alpha)$ por la probabili-

dad conjunta $\prod_{(u,v)\in\mathcal{W}_{(x,y)}} P(I_{(u,v)}|\alpha)$. De esta forma obtenemos el siguiente funcional de energía:

$$E\left[\Gamma,\{\alpha_i\}\right]=\sum_{i=1}^{M}\{\frac{\mu}{2}\oint_{\partial R_i} ds-\int\int_{R_i}\frac{1}{m}\int\int_{\mathcal{W}_{(x,y)}}\log P(I_{(u,v)}|\alpha_i)dudvdxdy+\lambda\}\ . \tag{3.10}$$

La ventana $\mathcal{W}$ soluciona los problemas antes expuestos. En primer lugar, se observa que cuanto mayor sea m mayor es la probabilidad de que la ventana sea representativa de la distribución y, por tanto, menor será el riesgo de error en la clasificación. Por otra parte, la ventana nos permite calcular estadísticas de mayor orden sobre $\{I_{(u,v)} : (u,v) \in \mathcal{W}_{(x,y)}$ respondiendo al segundo problema expuesto anteriormente. Sin embargo, la ventana no puede ser excesivamente grande o no podremos localizar los bordes de la imagen con precisión. Existe, por lo tanto, un compromiso entre el tamaño de la ventana y, como se demuestra, el nivel de señal frente al ruido. Nótese que la ecuación original de MDL (ecuación 3.1) se obtiene para el caso de $m = 1$. Se demuestra además que este tamaño mínimo es el óptimo para imágenes con altos niveles de ruido. Además, el uso de ventanas ayuda a reducir la sobresegmentación al suavizar los valores de media y varianza (ver [18]). En la figura 3.3 puede observarse este efecto.

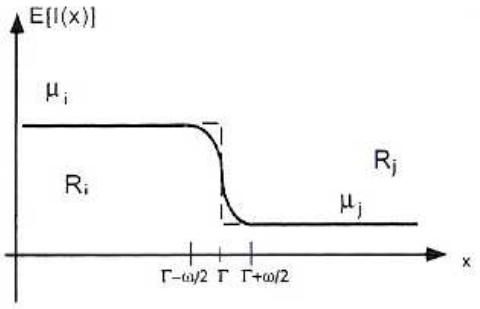


Figura 3.3.: Representación de la media de las distribuciones de los píxeles cercanos a la frontera Γ entre las regiones R_i y R_j. Usando el MDL estándar para distribuciones Gaussianas aparece un escalón en Γ (línea discontinua). Usando la ventana se suaviza el borde de forma que el valor de la media no cambia bruscamente de μ_i a μ_j en el intervalo que abarca la ventana, desde $\Gamma - \omega/2$ hasta $\Gamma + \omega/2$ (curva continua). Las varianzas se comportan de forma similar.

Pero, ¿cómo se relaciona el funcional de la ecuación 3.10 con el MDL estándar? Podemos reescribir esta ecuación de la forma:

$$E\left[\Gamma,\{\alpha_i\}\right]=\mu\oint_{\Gamma} ds-\int\int_{R}\sum_{i=1}^{M}\pi_{i(x,y)}\log P(I_{(x,y)}|\alpha_i)dxdy+\lambda M\ , \tag{3.11}$$

donde $\Gamma = \cup_{i=1}^{M}\partial R_i$ y $\pi_{i(x,y)} = \frac{\|R_i\cap\mathcal{W}_{(x,y)}\|}{m}$ tal que $\sum_{i=1}^{M}\pi_{i(x,y)} = 1$ para todo punto (x,y). En otras palabras, asumimos que cada píxel tiene una mezcla de distribuciones de la forma $P(I_{(x,y)}|\{\alpha_i\}) \propto \prod_{i=1}^{M} P^{\pi_i}(I_{(x,y)}|\alpha_i)$. Imaginemos una zona de la imagen donde hay dos regiones adyacentes definidas R_i y R_j. Si estamos evaluando un punto A dentro de R_i de forma que él

mismo y todos sus m píxeles vecinos están dentro de esa región, tendremos que $\pi_i = 1$, $\pi_j = 0$, $\forall j \neq i$. Un punto B que se encuentre cerca de la frontera entre las regiones R_i y R_j tendrá dentro de su ventana píxeles correspondientes a las dos distribuciones y los correspondientes π_i y π_j aparecerán en el multiplicatorio anterior.

Si asumimos distribuciones Gaussianas para $P(I_{(x,y)}|\alpha_i)$, entonces, la mezcla de distribuciones $P(I_{(x,y)}|\{\alpha_i\})$ sigue siendo una Gaussiana con medias y varianzas dependientes de la posición (x, y). Para los puntos en el interior de una región R_i las medias y varianzas serán iguales a las de la región. Para los puntos cercanos a las fronteras entre regiones R_i y R_j las medias y varianzas serán el resultado de las ponderaciones de las medias y varianzas de las dos regiones. De esta forma se suavizan los cambios entre las medias y varianzas al pasar de una región a otra y resulta un modelo mejor para imágenes reales, produciendo mejores resultados.

En el resto de esta sección exploraremos la ecuación 3.10 suponiendo distribuciones Gaussianas. Supongamos que tenemos la muestra producida por una ventana centrada sobre un punto (x, y), $\left\{I_{(u,v)}|(u, v) \in \mathcal{W}_{(x,y)}\right\}$, donde cada $I_{(u,v)} \sim N(\mu, \sigma^2)$ es mostrado en la figura 3.2(a). Podemos entonces calcular los parámetros $\bar{I}_{(x,y)}$ y $S^2_{(x,y)}$ como la media y varianza de la muestra obtenida con la ventana $\left\{I_{(u,v)}|(u, v) \in \mathcal{W}_{(x,y)}\right\}$ y obtener $\bar{I}_{(x,y)} \sim N(\mu, \sigma^2/m)$, que es la distribución representada por las líneas discontinuas en la figura 3.2(a). Como la varianza se ha dividido entre m, el riesgo de error en la clasificación se ha reducido en gran medida.

Podemos reescribir las fuerzas estadísticas generadas por una región R_i en el punto (x, y) en la dirección del vector $\vec{n}_i$ de la siguiente forma:

$$\begin{aligned}\frac{1}{m}\int\int_{\mathcal{W}_{(x,y)}} \log \Pr(I_{(u,v)}|\alpha_i)dudv =& \frac{1}{m}\log \prod_{(u,v)\in\mathcal{W}_{(x,y)}} \Pr(I_{(u,v)}|\mu_i, \sigma_i^2) \\ =& -\frac{1}{m}\left\{\log(2\pi\sigma_i^2)^{\frac{m}{2}} + \sum_{(u,v)\in\mathcal{W}_{(x,y)}} \frac{(I_{(u,v)} - \mu_i)^2}{2\sigma_i^2}\right\} \\ =& -\frac{1}{2}\left\{\log(2\pi\sigma_i^2) + \frac{(\bar{I} - \mu_i)^2}{\sigma_i^2} + \frac{S^2}{\sigma_i^2}\right\} .\end{aligned} \tag{3.12}$$

En la ecuación 3.12, el segundo término comprueba la media y el primero la varianza. Con esta "fuerza de varianza" añadida, pueden detectarse dos regiones con la misma media pero diferentes varianzas (ver figura 3.2). Por supuesto, la ecuación 3.12 puede generalizarse a mayores dimensiones para la segmentación de imágenes en color, por ejemplo.

Ahora, podemos obtener de nuevo la ecuación de movimiento del punto $\vec{v}$ sobre el borde $\Gamma_i \cap \Gamma_j$ a partir de la ecuación 3.10 si añadimos la ecuación 3.12 a la ecuación de movimiento 3.6.

$$\frac{d\vec{v}}{dt} = -\mu\kappa_{i(\vec{v})}\vec{\mathbf{n}}_{i(\vec{v})} - \frac{1}{2}\left\{\log\frac{\sigma_i^2}{\sigma_j^2} + \frac{(\bar{I}-\mu_i)^2}{\sigma_i^2} - \frac{(\bar{I}-\mu_j)^2}{\sigma_j^2} + \frac{S^2}{\sigma_i^2} - \frac{S^2}{\sigma_i^2}\right\}\vec{\mathbf{n}}_{i(\vec{v})} \ . \tag{3.13}$$

Antes de sintetizar el algoritmo RC completo al final de esta sección, explicaremos cómo se disminuye la energía fundiendo dos regiones adyacentes. supongamos que n_i, n_j, μ_i, μ_j, σ_i y σ_j son, respectivamente, el número del píxel, y la media y varianza de la intensidad de dos regiones R_i y R_j. Sea $R_{ij} = R_i \cup R_j$ la región resultante tras la fusión y $\ell_{ij} = \partial R_i \cap \partial R_j$ la frontera común entre R_i y R_j. Definimos además $\alpha_{ij} = (\mu_{ij}, \sigma_{ij})$ como la media y varianza de R_{ij}. Se comprueba fácilmente que:

$$\mu_{ij} = \frac{1}{n_i + n_j}(n_i\mu_i + n_j\mu_j) \ ,$$

$$\sigma_{ij}^2 = \frac{1}{n_i + n_j}\{n_i\sigma_i^2 + n_j\sigma_j^2 + \frac{n_i n_j}{n_i + n_j}(\mu_i - \mu_j)^2\} \ .$$

Luego el incremento ΔE en la energía al unir las regiones R_i y R_j es:

$$\begin{aligned}\Delta E =& -\mu\int_{l_{ij}} ds - \int\int_{R_{ij}} \frac{1}{m}\int\int_{\mathcal{W}_{(x,y)}} \log\Pr(I_{(u,v)}|\alpha_{ij})dudvdxdy \\ &+ \int\int_{R_i} \frac{1}{m}\int\int_{\mathcal{W}_{(x,y)}} \log\Pr(I_{(u,v)}|\alpha_i)dudvdxdy \\ &+ \int\int_{R_j} \frac{1}{m}\int\int_{\mathcal{W}_{(x,y)}} \log\Pr(I_{(u,v)}|\alpha_j)dudvdxdy - \lambda \\ =& -\mu|\partial R_i \cap \partial R_j| - \lambda + \frac{1}{2}(n_i\log\frac{\sigma_{ij}^2}{\sigma_i^2} + n_j\log\frac{\sigma_{ij}^2}{\sigma_j^2} + 1) \ .\end{aligned} \tag{3.14}$$

Si $\Delta E \leq 0$ la fusión de R_i y R_j hará decrecer la energía.

3.1.4. El algoritmo

En las secciones anteriores se ha descrito la estrategia de segmentación de competición de regiones de forma analítica. En estas secciones se ha comentado que dada la descripción del funcional de energía (ecuación 3.1) y su dependencia de dos grupos de variables, parece lo más apropiado secuenciar el proceso de minimización en dos pasos que se suceden alternativamente. Primero se fijan los bordes y se calculan los descriptores α_i para cada región. En un segundo paso se fijan los descriptores y se mueven los bordes de forma que las regiones adyacentes *compiten* por los píxeles cercanos a su frontera. Estos dos pasos que se repiten iterativamente forman la primera

etapa del algoritmo, mediante la que se alcanza un mínimo local en el funcional de energía manteniendo constante el número de regiones. No obstante, al existir la posibilidad de que un número menor de regiones implique un descenso en la energía, se incluye una segunda etapa en la que se produce la fusión de dos regiones siempre y cuando se produzca tal decremento.

Este proceso, expuesto en forma de algoritmo es el siguiente:

1. Inicializar la segmentación. Podemos poner N semillas distribuidas aleatoriamente a lo largo de la imagen y toda el área de fondo (el área no ocupada por ninguna región semilla) se trata como una única región con distribución de probabilidad uniforme.
2. Fijar la segmentación Γ y calcular los parámetros $\{\alpha_i\}$ maximizando $P(I : \alpha_i)$.
3. Fijar $\{\alpha_i\}$ y mover los bordes Γ minimizando la función de energía. Cuando dos regiones semilla se encuentran se forma una frontera entre ellas y ambas regiones compiten a lo largo de ella.
4. Ejecutar los pasos 2 y 3 iterativamente haste que el movimiento de Γ converja. Después ir al paso 5.
5. Si hay región de fondo sin ocupar por ninguna región semilla poner una nueva semilla en la región de fondo e ir al paso 2; en caso contrario, ir al paso 6.
6. Fundir dos regiones adyacentes de forma que se produzca el mayor descenso de energía posible, ir al paso 2. Si no es posible reducir la energía mediante la fusión de regiones, ir al paso 7.
7. Parar.

3.2. RC en nuestra aplicación, descripción del algoritmo

En esta parte de la memoria describiremos la adaptación y optimización del algoritmo RC para la segmentación de fotogramas estereoscópicos. En primer lugar estableceremos el entorno de trabajo del proyecto, encuadrándolo dentro del sistema de visión artificial para la extracción de parámetros de los músculos artificiales. Después describiremos de forma general la actuación del algoritmo adaptado y, en las siguientes subsecciones analizaremos los puntos más interesantes de nuestra adaptación. En último lugar se indican las fases de desarrollo por las que pasó la aplicación hasta llegar a un estado satisfactorio de resultados y rendimiento.

Antes de comenzar con la descripción del entorno de trabajo en el que se encuadra este proyecto (sistema para la extracción de parámetros de los músculos artificiales) subrayaremos que

el objetivo final de este proyecto no es solamente la creación de un algoritmo capaz de cumplir los requisitos mínimos para la caracterización de los polímeros activos sino la creación de un algoritmo de segmentación robusto y eficiente aplicable a otros tipos de imágenes.

3.2.1. Entorno de trabajo

Los evolución de los materiales inteligentes, es decir, materiales capaces de responder de modo reversible y controlable ante diferentes estímulos físicos o químicos externos, ha sido espectacular en los últimos años. Dentro de estos materiales activos destacan los polímeros electroactivos por el interés que despiertan sus posibles aplicaciones. Una de las aplicaciones más interesantes de estos materiales son los llamados actuadores o músculos artificiales. Estos dispositivos, todavía en fase de investigación y desarrollo, podrían ser utilizados en multitud de ámbitos tan dispares como prótesis para extremidades humanas o la creación de sistemas motrices para vehículos de exploración de otros planetas.

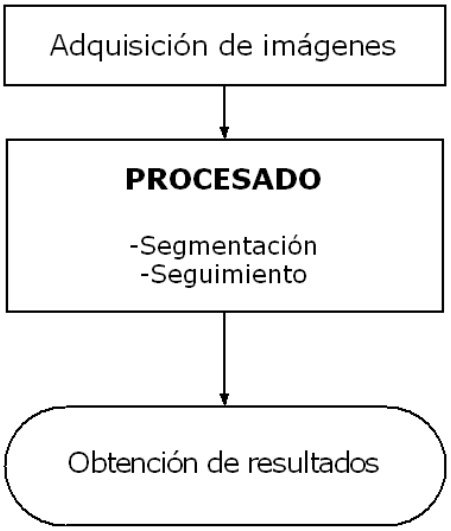


Figura 3.4.: Método para extracción de parámetros.

Para la observación y caracterización mecánica de estos músculos se ha desarrollado un método basado en visión artificial y procesado de imagen. Usando dos cámaras digitales (ver figura 3.5) y mediante técnicas de visión estereoscópica es posible la caracterización del músculo en 3D.

Para esta caracterización es fundamental el procesado digital de las imágenes obtenidas por las cámaras. Este procesado consiste en la segmentación del músculo y el seguimiento de éste a lo largo de su recorrido.

El proceso completo para la caracterización de los músculos es el siguiente:

- *Adquisición*. Control de las cámaras y adquisición de imágenes estéreo.
- ***Procesado de las imágenes***. Compuesto por dos tareas:
  - Segmentación de la imagen. Esta tarea se encarga de aislar el músculo del resto de la imagen.
  - Seguimiento del objeto.
- *Resultados*. Extracción de parámetros: movimiento, energía de curvatura, etc.

Como se puede ver en el diagrama de la figura 3.4, este proyecto se centra en el bloque dedicado a la *segmentación* y *seguimiento* del músculo a través de la secuencia de fotogramas que le proporciona el bloque de *adquisición* formado por las dos cámaras dispuestas ortogonalmente respecto al músculo en movimiento.

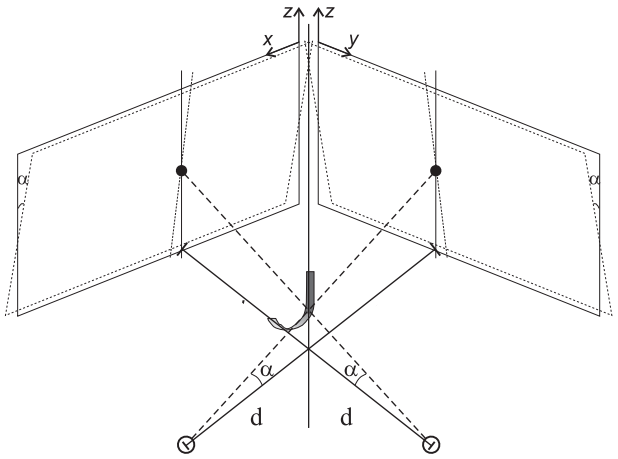


Figura 3.5.: Disposición de las cámaras para la obtención de los fotogramas estereoscópicos.

El sistema de adquisición es el encargado de generar la entrada de datos al bloque de procesado en forma de pares de fotogramas. El algoritmo de segmentación y seguimiento recibe estas secuencias de imágenes en formato JPEG (aunque es capaz de trabajar con cualquiera de los formatos digitales más comunes) numeradas secuencialmente. Las dimensiones de las imágenes recibidas pueden variar de unas secuencias a otras. Generalmente trataremos con imágenes cuadradas de 120×120 ó 150×150 píxeles aunque el tamaño y la forma de éstas no supone ninguna restricción para el funcionamiento del algoritmo. Las imágenes generadas por el sistema de adquisición son en color y, como veremos más adelante, el algoritmo de segmentación y seguimiento está diseñado para trabajar sobre imágenes a 255 niveles de grises. No obstante, es susceptible de ampliación para el funcionamiento con imágenes RGB mediante el incremento de la dimensionalidad de los descriptores e, inevitablemente, del coste computacional.

Figura 3.6.: Interfaz diseñado para la captura de los fotogramas en el sistema de adquisición.

La salida de datos del bloque de segmentación y seguimiento debe ser útil para la extracción de los parámetros del músculo. Esta salida consiste en otra secuencia de fotogramas igual a la de entrada sobre la que se superpone el resultado de la segmentación, es decir, el contorno del músculo artificial.

El bloque de *obtención de resultados* es el encargado de cuantificar parámetros de interés tales como el movimiento y la energía de curvatura a partir de la segmentación de cada fotograma. Esta parte del sistema es ajena a este proyecto.

3.2.2. Descripción general del algoritmo

En esta sección se describe brevemente el funcionamiento del algoritmo encargado de la segmentación y seguimiento del músculo a lo largo de la secuencia de fotogramas estereoscópicos. En secciones posteriores entraremos en detalle sobre diferentes aspectos de interés que merece la pena desarrollar un poco más.

En la sección anterior hemos visto el entorno de trabajo que envuelve a nuestra aplicación y se ha definido el flujo de entrada y salida de información. A la entrada de nuestro algoritmo tenemos una secuencia de fotogramas estereoscópicos y a la salida se genera otra secuencia correspondiente a la segmentación de la entrada. En la figura 3.7 podemos ver el flujo de entrada y salida del algoritmo.

Para conseguir la segmentación que se muestra en la figura 3.7 se ha programado un algoritmo en lenguaje *Matlab* (*Mathworks Inc.* ©) formado por diferentes rutinas y subrutinas cada una encargada de realizar una función específica.

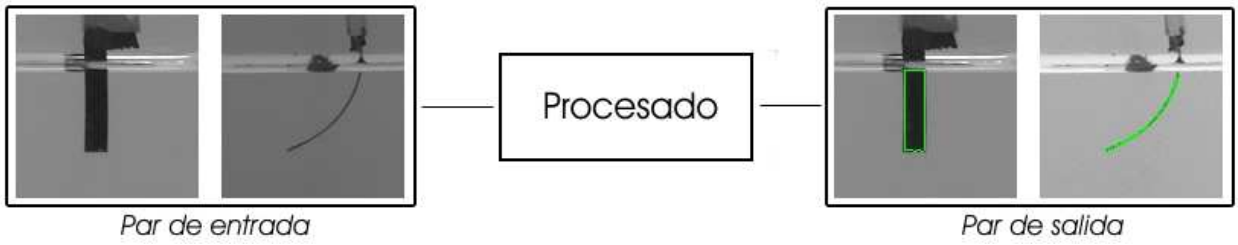
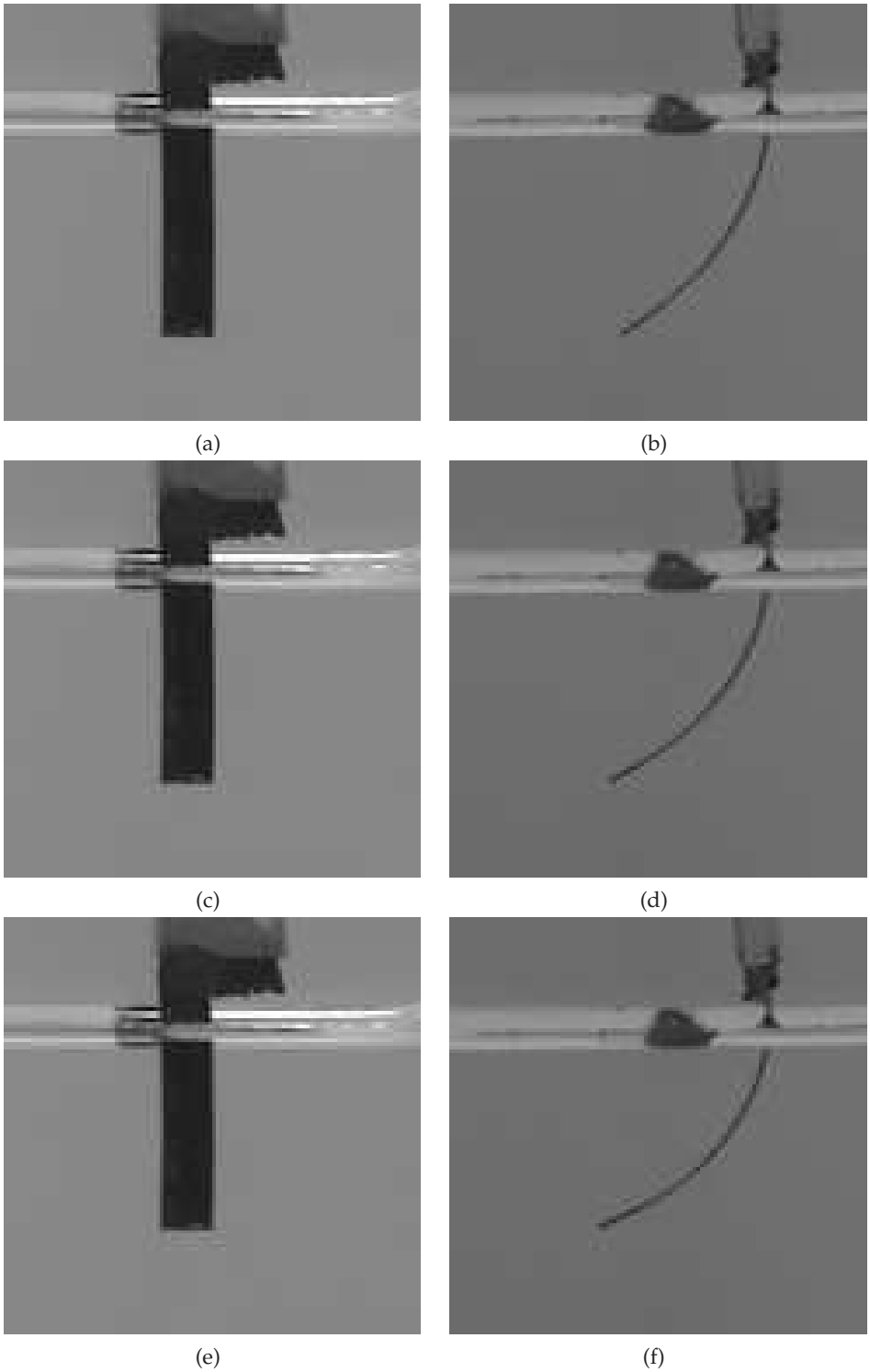


Figura 3.7.: Flujo de entrada y salida de la aplicación.

Como ya hemos comentado, la aplicación está estrechamente relacionada con el algoritmo de segmentación RC de Zhu y Yuille [1] y muchos de los elementos presentes en ella son comunes a RC. Otras características de la aplicación han sido adaptadas a la naturaleza del caso especial que nos ocupa.

Haremos la descripción de nuestra aplicación de forma secuencial. Es decir, partiendo de nuestra entrada (la secuencia de fotogramas captados por el sistema de cámaras) hasta llegar al resultado de salida. Tenemos una serie de pares de imágenes que van "entrando" al algoritmo en el mismo orden que han sido captadas. Llamaremos $A(t)$ al fotograma que capta la imagen del músculo sobre el plano XZ (ver el sistema de coordenadas de la figura 3.5) y $B(t)$ al fotograma captado en el mismo instante de tiempo en el plano YZ. De forma que la secuencia de fotogramas que conforma la entrada sigue el siguiente orden $S(t) = A(0), B(0); A(1), B(1); \ldots; A(t_{max}), B(t_{max})$ donde t_{max} representa el instante de tiempo en el que se tomó el último par de fotogramas de la secuencia. En la figura 3.8 podemos ver un ejemplo de estas secuencias de entrada. Como puede verse, las variaciones entre los fotogramas consecutivos son muy pequeñas, lo que facilitará el seguimiento del músculo. Además, puede verse en la figura que las cámaras están perfectamente calibradas para los extremos del músculo se encuentren en la misma posición relativa en cada par de fotogramas. Esta situación es importante para el uso de la información entre unos fotogramas y otros.

El procesado de los fotogramas se realiza a pares $A(t), B(t)$. Esto es así porque la información contenida en un fotograma de uno de los planos, por ejemplo XZ, es útil para la segmentación del fotograma del otro plano YZ. Como veremos más adelante, no sólo se usa información entre los pares de fotogramas de un mismo instante de tiempo t sino también entre los fotogramas de instantes de tiempo consecutivos t y $t-1$. Sin embargo, en la inicialización de nuestra aplicación $t = 0$ sólo contamos con el par de fotogramas $A(0)$ y $B(0)$. La primera fase es, por tanto, la segmentación inicial de estos fotogramas. Como se ha comentado anteriormente, el propósito de la aplicación es la segmentación de secuencias de fotogramas de músculos artificiales pero sin pérdida de generalidad para la segmentación de otros tipos de imágenes con un mínimo ajuste. Por esta razón, la segmentación del primer par de fotogramas se realiza con el mínimo de información a priori y siguiendo el algoritmo definido por RC (ver sección 3.1.4).



(a) (b) (c) (d) (e) (f)

Figura 3.8.: Secuencia $S(t)$ de fotogramas. (a) Fotograma $A(0)$ del plano XZ . (b) Fotograma $B(0)$ del plano YZ. (c) Fotograma $A(1)$. (d) Fotograma $B(1)$. (e) Fotograma $A(2)$. (f) Fotograma $B(2)$.

El número de semillas empleadas para iniciar la competición de regiones es totalmente configurable en nuestra aplicación y el número óptimo de éstas depende de la imagen a segmentar. En cualquier caso, la segmentación es satisfactoria independientemente del número de semillas iniciales. En primer lugar se realiza la segmentación del fotograma $A(0)$, ya que corresponde a la imagen en que el objeto (músculo) ocupa más área, facilitándose así el crecimiento de la región objetivo. Esto mejora un poco la velocidad de la segmentación aunque tampoco determina su éxito ya que en el caso complementario (segmentación del fotograma $B(0)$) también se consigue la detección de la silueta del músculo.

Una vez que el algoritmo de segmentación ha realizado su trabajo, esto es, todo el espacio de la imagen $I(x, y)$ está dividido en N regiones R_i, se requiere la interacción del usuario para seleccionar la región de interés, es decir, la aplicación pide que se identifique el objeto que ha de segmentar y sobre el que ha de realizar el seguimiento. Tras realizar la segmentación del fotograma $A(0)$ se usa la información obtenida para segmentar el fotograma $B(0)$. La segmentación del primer fotograma del plano YZ es más rápida que la del XZ gracias a esta información ya que el algoritmo "sabe" lo que está buscando y es capaz de ignorar a los demás elementos del fondo. El trasvase de información y el aprovechamiento de ésta se realiza mediante la aplicación de técnicas de morfología matemática que acotan las regiones de actuación del algoritmo de segmentación.

Llegados a este punto, ya disponemos de la segmentación Γ de los dos fotogramas iniciales $A(0)$ y $B(0)$. A partir de este momento, el algoritmo entra en un régimen de funcionamiento *estacionario* que no abandonará hasta el final de la ejecución. Esto se debe a que ahora ya disponemos de la información a priori obtenida a partir de la segmentación inicial. Durante este régimen de funcionamiento que hemos llamado *estacionario* se realiza la segmentación de los fotogramas de la secuencia $S(t) = A(1), B(1); A(2), B(2); \ldots; A(t_{max}), B(t_{max})$. Para la segmentación de estos fotogramas ya no se aplica el algoritmo RC descrito en la sección 3.1.4 de forma estricta porque ya no es necesario. En lugar de esto se realiza una segmentación mucho más rápida en la que el paso de fusión de regiones sólo se realiza en ocasiones excepcionales reduciéndose así el tiempo de proceso. Además, durante todo este régimen de funcionamiento el número de regiones permitidas se reduce a dos (target y background) durante la mayor parte del tiempo de forma que reducimos la complejidad y el cómputo.

El seguimiento del movimiento del músculo a lo largo de la secuencia se realiza mediante técnicas de segmentación espacial (ver sección 2.7.1) combinadas con un preprocesado y postprocesado de los fotogramas mediante operadores morfológicos (ver anexo B). Este seguimiento se basa en el cálculo de la diferencia entre las imágenes $A(t)$ y $A(t-1)$ para hallar aquellos píxeles que se han movido.

La figura 3.9 muestra el modo en que la aplicación segmenta una secuencia de vídeo estereoscópico de tres pares de fotogramas. En primer lugar se realiza la segmentación del primer

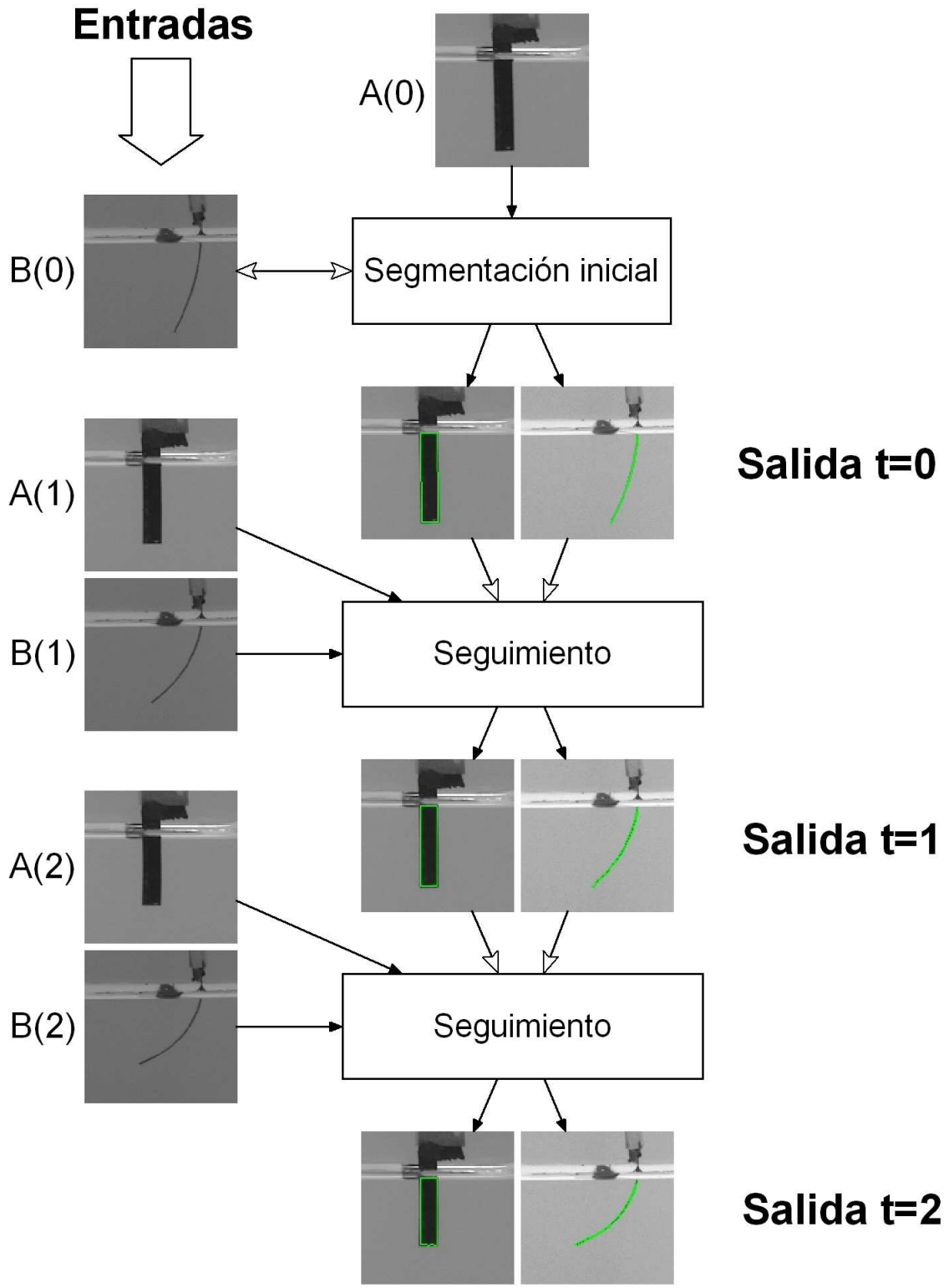


Figura 3.9.: Segmentación de una secuencia de 3 pares de fotogramas. Las flechas con la punta blanca indican trasvase de información entre fotogramas.

fotograma del plano XZ mediante el algoritmo RC. La información obtenida acerca de la posición del músculo en el instante $t = 0$ sirve para optimizar la segmentación del fotograma del plano YZ. Una vez detectado el músculo, la aplicación se centra en el seguimiento del objeto a través de los fotogramas siguientes.

Adicionalmente, se han implementado unas funciones de monitorización del proceso de segmentación y seguimiento que sirven para observar la evolución de los descriptores α_i de las regiones R_i existentes además del centro de masas del objeto de interés. La salida se presenta en forma de fotogramas con el resultado de la segmentación superpuesta y en formato de video AVI.

3.2.3. Segmentación inicial

En la sección anterior comentamos la importancia de la segmentación de los fotogramas iniciales, distinguiéndola del proceso del resto de fotogramas subsiguientes. En efecto, la segmentación de los dos primeros fotogramas $A(0)$ y $B(0)$ es crítica para el correcto funcionamiento del algoritmo ya que es en esta etapa donde se detecta el objeto de interés y se obtiene la información necesaria para la segmentación de las siguientes imágenes. Es particularmente importante la segmentación del fotograma del plano XZ, $A(0)$. Es sobre esta primera imagen donde se aplica el algoritmo de Zhu con más rigor ya que la información a priori de la que disponemos es mínima.

El proceso de segmentación del fotograma $A(0)$ comienza tal como indica el algoritmo de RC de la sección 3.1.4. El primer paso consiste en distribuir N semillas aleatoriamente por el dominio de la imagen $I(x, y)$ (ver figura 3.10(b)). Tanto el número de semillas, N, como el tipo de distribución de éstas (aleatorio o a elección del usuario) son parámetros configurables de la aplicación. La implementación final de nuestro algoritmo permite el uso de un número arbitrario semillas en la segmentación inicial de forma que pueden segmentarse imágenes con varios objetos u objetos con diferentes partes o texturas de forma más eficiente. Si las semillas son situadas en un punto (x, y) a voluntad del usuario y con cierto criterio, esto es, poniendo las semillas en el centro de zonas homogéneas de la imagen el rendimiento del algoritmo es mayor. En cualquier caso, se genera una segmentación Γ correcta siempre que los descriptores usados proporcionen al algoritmo la información suficiente para distinguir entre las diferentes regiones.

El tamaño de las semillas iniciales es de 1 píxel. Sin embargo el cálculo de los descriptores α_i para estas semillas iniciales se hace sobre una ventana de muestreo de 5×5 píxeles de tamaño. El uso de esta ventana previene contra el posible ruido presente en la imagen: si una de las semillas iniciales cayera sobre un píxel (x, y) cuyo valor no sea representativo de la zona de la imagen en que se encuentra, esta semilla no podría crecer ya que su descriptor sería tan diferente de sus vecinos que el test de verosimilitud le impediría expandirse. Dicho de otra forma, las fuerzas estadísticas de su entorno impedirían que esta región inicial se expandiese. Por otra parte, el

tamaño de la ventana no puede ser arbitrariamente grande ya que si ésta abarcase algún borde significativo de la imagen el descriptor generado podría no ser representativo de ninguna de las dos zonas colindantes y el crecimiento de la región no sería el esperado. El tamaño óptimo de esta ventana de muestreo inicial depende, en realidad, de la relación señal/ruido de la imagen, en [1] puede encontrarse una discusión acerca de las semillas y el tamaño de las ventanas de muestreo. Uno de los aspectos más favorables de la Competición de Regiones es que las "buenas" semillas se imponen sobre las "malas" mediante el propio proceso de competición (ver figura 3.10). Sin embargo, si las semillas iniciales no son buenas, el tiempo de convergencia del algoritmo aumenta considerablemente.

Tras colocar las N semillas iniciales y asignarles un descriptor representativo de la zona en que se encuentran, tenemos una primera aproximación de la segmentación del fotograma. Esto es, una matriz de dos dimensiones $m \times n$ que coincide con las dimensiones en píxeles del fotograma. Esta matriz la llamamos $\Gamma_{A(0)}$ y está formada por ceros en su totalidad, excepto en los N puntos (x, y) donde están las semillas iniciales, cada una de ellas etiquetada por un identificador de región R_i. Además a cada semilla/región se le asigna un descriptor estadístico α_i que caracteriza a esa región y sirve para controlar el crecimiento de ésta.

El siguiente paso en la segmentación inicial es el crecimiento de estas N regiones. Este crecimiento se ha implementado mediante operadores morfológicos que dilatan (ver anexo B) las semillas usando un elemento estructurante de 3×3 píxeles de forma que las regiones crecen a lo largo de todo su perímetro en cada iteración. Los nuevos píxeles producidos por esta dilatación se comparan con el descriptor α_i de la región R_i y sólo son anexionados a ésta si se ajustan a la distribución $P(I|\alpha_i)$ definida por nuestro criterio de homogeneidad. En el caso más simple de nuestra implementación para las imágenes de músculos artificiales, el descriptor es la media y la decisión de anexionar o no un nuevo píxel se toma mediante un umbral $T_{crecimiento}$ absoluto. En cada iteración de crecimiento los descriptores α_i se recalculan de forma que siempre son representativos del área que cubre cada región R_i. Cuando los valores de nuestra matriz $\Gamma_{A(0)}$ ya no varían, es decir, el crecimiento de regiones converge, se comprueba que todas las posiciones dentro de $\Gamma_{A(0)}$ tienen algún valor distinto de cero. O lo que es lo mismo, todos los puntos de la imagen $I(x, y)$ han sido asignados a alguna región. Si todavía quedan puntos sin asignar, se coloca una nueva semilla y se repiten los pasos anteriores.

Según el algoritmo RC, se fundirán todas aquellas regiones adyacentes que provoquen un descenso de la energía. En nuestra implementación esto se traduce en la fusión de aquellas regiones vecinas cuyos descriptores tengan valores similares (figuras 3.10(g) y (h)). Dadas dos regiones adyacentes R_i y R_j con descriptores α_i y α_j respectivamente se realizará la fusión de ambas cuando la diferencia entre sus descriptores sea menor que un umbral T_{fusion} definido por el usuario.

En el caso que nos ocupa y, en general, la elección de los umbrales es siempre determinante

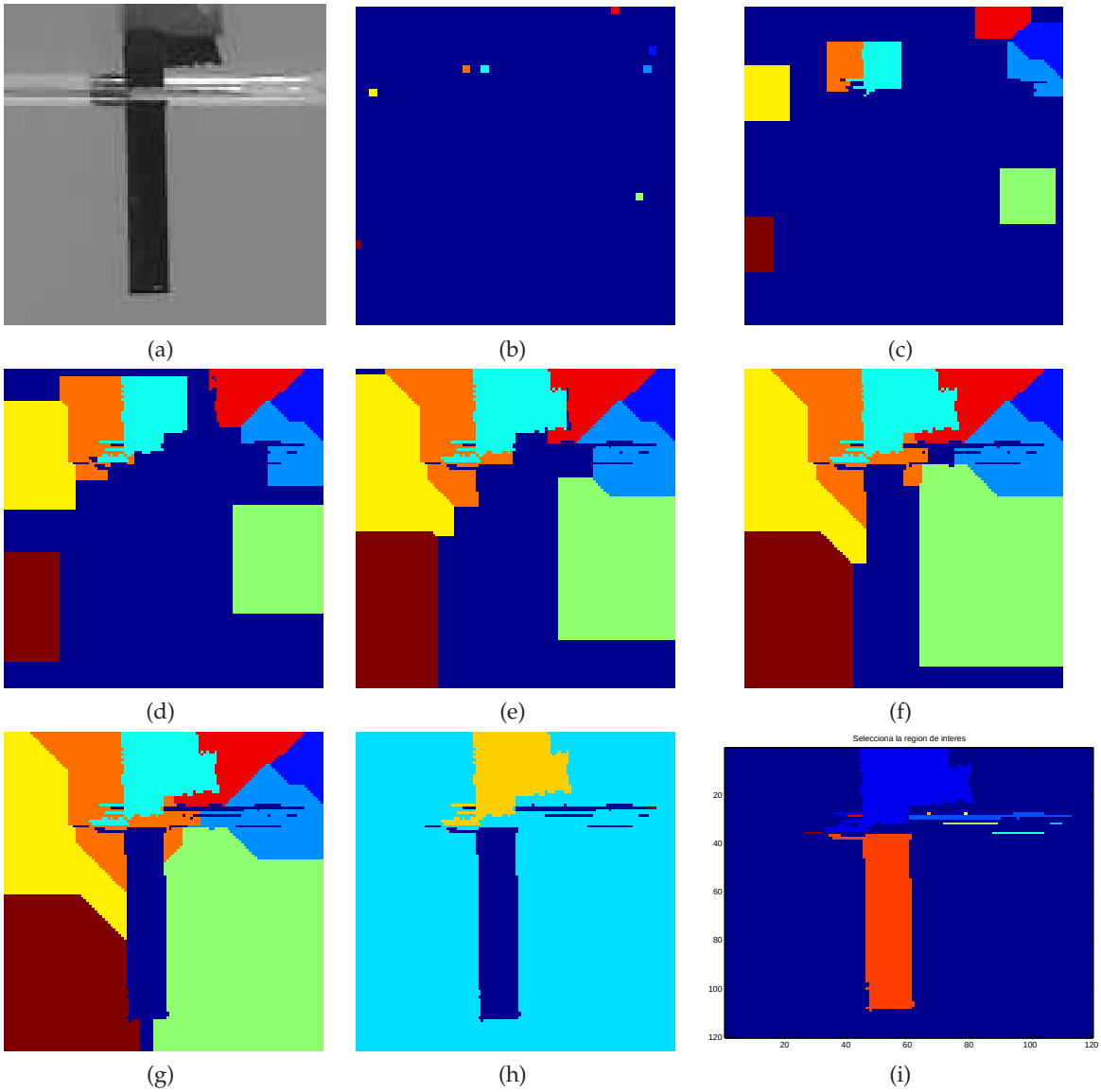


Figura 3.10.: Segmentación inicial del fotograma $A(0)$ con $N = 8$ semillas dispuestas aleatoriamente (a) Fotograma $A(0)$. (b) Segmentación $\Gamma_{A(0)}$ tras la primera iteración de crecimiento; (c), (d), (e), (f), (g) y (h) representan la evolución de $\Gamma_{A(0)}$ en intervalos de 10 iteraciones. (i) Segmentación $\Gamma_{A(0)}$ final tras un total de 160 iteraciones. La aplicación pide la selección de la región de interés por parte del usuario.

respecto a los resultados obtenidos. Si los umbrales $T_{crecimiento}$ y T_{fusion} son muy restrictivos las regiones se volverán demasiado reticentes a la hora de anexionarse nuevos píxeles o fundirse con otras regiones. Esto provocará la creación de más regiones y, además de ralentizar el algoritmo, puede llevar a la sobresegmentación de la imagen. Si por el contrario los umbrales están demasiado relajados, corremos el riesgo de perder regiones con significado en la imagen. La elección de estos umbrales determinará en gran medida el resultado de la segmentación y deben ser elegidos cuidadosamente. En la sección 2.4.3 se exponen algunas consideraciones interesantes acerca de la elección de umbrales.

La figura 3.10 ilustra el proceso de competición de regiones. Cada semilla inicial (en este ejemplo han sido dispuestas aleatoriamente) comienza a crecer hasta que topa con otra región o hasta que encuentra una zona de la imagen que no coincide con su descriptor. A pesar del ruido introducido en la imagen por la superficie del agua en la que se sumerge el músculo, la segmentación final es correcta. Sin embargo, el rendimiento del algoritmo se ve ligeramente penalizado por este ruido ya que se crean pequeñas regiones en la superficie del agua. Esta es la razón de que el proceso de la figura 3.10 necesite tantas iteraciones para converger al resultado final. Este problema en la segmentación inicial puede resolverse haciendo uso del conocimiento a priori de la posición del agua respecto al músculo.

Cuando todas las fusiones pertinentes hayan tenido lugar y no queden puntos con valor cero en $\Gamma_{A(0)}$, la segmentación del primer fotograma $A(0)$ se da por terminada. En este punto, la aplicación pide la intervención del usuario para seleccionar por medio del ratón la región de interés (figura 3.10(i)). Puede prescindirse de esta intervención del usuario cuando el algoritmo se esté usando exclusivamente para la segmentación de imágenes de músculos artificiales ya que la posición inicial de éstos es conocida en la mayoría de los casos. No obstante, es una opción útil para la segmentación de otros tipos de imágenes y el testeo del algoritmo en diferentes situaciones.

3.2.4. Trasvase de información entre pares de fotogramas ortogonales

Uno de los aspectos más favorables de nuestra aplicación es el uso que hace de la información para segmentar el primer par de fotogramas iniciales así como los fotogramas sucesivos en tiempo. El hecho de contar con dos cámaras estereoscópicas situadas ortogonalmente entre sí permite usar la información de una imagen para acelerar la segmentación de la otra. Además, al trabajar con secuencias de vídeo, el trasvase de información entre fotogramas consecutivos es de gran ayuda para el seguimiento del objeto de interés.

Como hemos comentado anteriormente, la aplicación sólo usa la información de un plano para la segmentación del plano ortogonal en el primer par de fotogramas (ver figura 3.9), esto es, entre $A(0)$ y $B(0)$. En los siguientes instantes de tiempo, el seguimiento del objeto en $A(t)$ sólo

usa información de $A(t-1)$ y lo mismo para el plano YZ con $B(t)$ y $B(t-1)$. Nada impide que se implemente el trasvase de información entre cada par de fotogramas ortogonales $A(t)$ y $B(t)$, práctica que puede resultar útil en secuencias de imágenes de otro tipo. No obstante, los experimentos realizados con las secuencias de músculos artificiales no muestran mejorías relevantes al incluir esta característica. Por esta razón, el flujo de información $A(t) \longrightarrow B(t)$ sólo está implementado para $t = 0$, liberando así a la aplicación de un trabajo extra durante la segmentación del vídeo.

En esta sección describimos cómo se traslada la información obtenida de la segmentación $\Gamma_{A(t)}$ al proceso de segmentación del fotograma $\Gamma_{B(t)}$ acotando el rango de búsqueda en $B(t)$.

Partimos de la segmentación $\Gamma_{A(t)}$ (que suponemos correcta) y queremos obtener la segmentación del fotograma complementario $\Gamma_{B(t)}$ que, de momento, está sin inicializar. La información que podemos usar de la segmentación $\Gamma_{A(t)}$ es la siguiente:

- La posición del objeto de interés $R^{A(t)}_{interes}$ en la imagen $A(t)$
- El valor $\alpha^{A(t)}_{interes}$ del descriptor del objeto de interés $R^{A(t)}_{interes}$ en $A(t)$

La posición del objeto de interés en $A(t)$ la usaremos para acotar la búsqueda de éste en $B(t)$. Como la posición de las cámaras está perfectamente calibrada para que las imágenes de ambos planos aparezcan a escala y en la misma posición relativa, podemos suponer que las coordenadas en z (eje vertical) de los puntos que forman $R^{B(t)}_{interes}$ serán siempre las mismas que en $R^{A(t)}_{interes}$. No podemos decir lo mismo de las coordenadas en el eje horizontal x. La forma y el sentido de desplazamiento del músculo en los fotogramas del plano XZ difiere bastante a la forma y desplazamiento vistos desde el plano YZ (ver figura 3.8). Para acotar la búsqueda en las coordenadas x del fotograma $B(t)$ se usa el descriptor $\alpha^{A(t)}_{interes}$ para encontrar los puntos de $B(t)$ dentro del margen acotado en y que más se parecen a la región de interés de $A(t)$. Para prevenir la presencia de ruido en $B(t)$ y compensar las diferencias de iluminación entre $A(t)$ y $B(t)$ se aplica un desplazamiento al valor de $\alpha^{A(t)}_{interes}$ al aplicarlo sobre la imagen $B(t)$. Además, una vez acotada el área donde se espera encontrar el músculo, ésta se dilata morfológicamente para prevenir posibles imprecisiones en la elección del desplazamiento sobre el descriptor $\alpha^{A(t)}_{interes}$. Todos los puntos que queden fuera de los márgenes serán etiquetados directamente como fondo, de forma que el proceso de segmentación se simplifica notablemente.

Llegado a este punto, el algoritmo comienza la segmentación del fotograma $B(t)$ por el método de competición de regiones aunque, en realidad, dicha competición no se produce casi nunca ya que más del $90\,\%$ del área de $B(t)$ está ya segmentada gracias a la información obtenida de $\Gamma_{A(t)}$. No obstante, el algoritmo procede a la ejecución de RC para prevenir ciertas situaciones

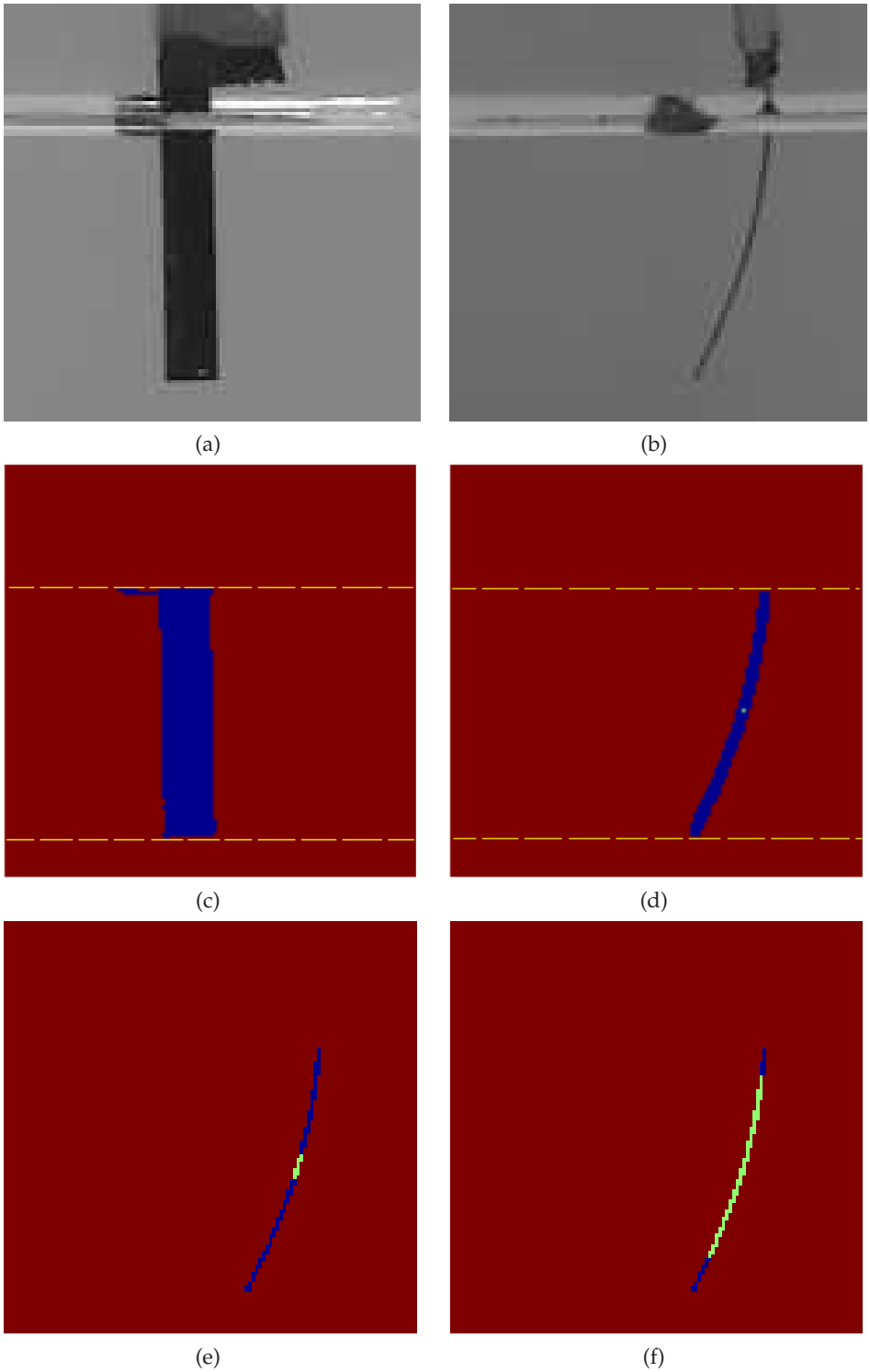


Figura 3.11.: Trasvase de información entre fotogramas ortogonales. (a) Fotograma $A(t)$. (b) Fotograma $B(t)$. (c) Segmentación $\Gamma_{A(t)}$. (d) Acotación del rango de búsqueda en $B(t)$. Se coloca una semilla en la coordenada y correspondiente al centro de masas de $R^{A(t)}_{interes}$ y la coordenada x con mayor verosimilitud respecto a $\alpha^{A(0)}_{interes}$. (e) y (f) muestran el crecimiento de regiones.

adversas. Por ejemplo, si el músculo está tan curvado que su extremo sobresale por encima de la superficie del agua la región de interés en el plano YZ aparecerá seccionada por el reflejo del agua. En esta situación el algoritmo también debe ser capaz de encontrar la posición de toda el área del músculo y necesitará de la robustez que le presta la competición de regiones.

Hemos dicho que, tras la acotación del área de búsqueda, se procede con la segmentación por competición de regiones. Esto es, plantar semillas y esperar que crezcan y compitan entre sí. En este punto de la segmentación del fotograma $B(t)$ en que hemos acotado la zona de búsqueda no tiene sentido distribuir las semillas aleatoriamente por toda la imagen. En lugar de eso, se planta una semilla en el centro de la zona acotada y se espera que crezca hasta completar la segmentación $\Gamma_{B(t)}$ (figura 3.11(d)). Más concretamente, las coordenadas de la semilla también se eligen usando la información de la segmentación $\Gamma_{A(t)}$. La semilla se planta en la coordenada z correspondiente al centro de masas de $R^{A(t)}_{interes}$ y la coordenada x con mayor verosimilitud respecto a $\alpha^{A(0)}_{interes}$. Esta precisión a la hora de situar la semilla inicial es importante para evitar que el proceso de crecimiento y competición de regiones se alargue. Si la semilla cayera en una zona correspondiente al fondo de la imagen, el algoritmo tendría que poner más semillas y entrar en el proceso de fusión de regiones que es, precisamente, lo que tratamos de evitar mediante el trasvase de información entre fotogramas.

La figura 3.11 ilustra el proceso que acabamos de describir. Las figuras (a) y (b) muestran los fotogramas $A(t)$ y $B(t)$ respectivamente. La figura 3.11 es la segmentación $\Gamma_{A(t)}$ de la que el algoritmo extrae la información para acelerar la segmentación de $B(t)$. Las figuras (d), (e) y (f) muestran la posición y crecimiento de la semilla inicial.

3.2.5. Segmentación del movimiento: Seguimiento

En esta sección veremos cómo se aprovecha la información obtenida en la segmentación de los fotogramas $A(t-1)$ y $B(t-1)$ para acelerar la segmentación de los fotogramas siguientes $A(t)$ y $B(t)$. La explicación que sigue a continuación hace referencia a los fotogramas del plano YZ y nos referiremos, exclusivamente, a las imágenes $B(t)$ y $B(t-1)$. No obstante, la segmentación de los fotogramas $A(t)$ es totalmente análoga.

Partiendo de la segmentación $\Gamma_{B(t-1)}$, que suponemos correcta, queremos segmentar el siguiente fotograma $B(t)$. Para ello usaremos la técnica para segmentación de vídeo descrita en la sección 2.7.1 que se basa en obtener la imagen diferencia de resultante de la operación $I(t)-I(t-1)$ (ver ecuación 2.36), donde $I(t)$ e $I(t-1)$ son dos imágenes adyacentes en el tiempo. Nuestra aplicación usa este método pero, en lugar de aplicarlo sobre todo el dominio de la imagen se aplicará solamente sobre los alrededores de la región de interés. Para ello, tomamos como verdadera la suposición de que la velocidad del movimiento es tal que si el objeto de interés se encuentra en una

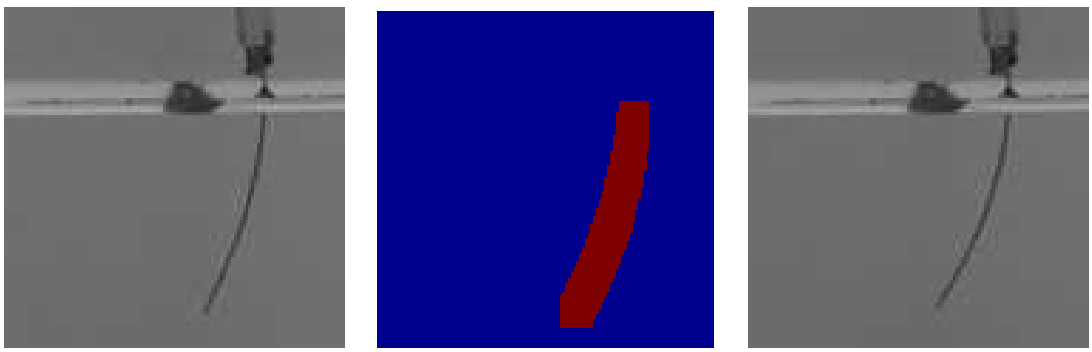
posición determinada en el fotograma $B(t-1)$, su posición en el fotograma $B(t)$ se ubicará en los alrededores de su posición anterior. Los músculos artificiales son dispositivos capaces de realizar movimientos con una velocidad limitada y, generalmente, constante. Esto nos permite, con cierta probabilidad de éxito, predecir la posición del músculo en el instante t si conocemos la posición que ocupaba en el instante $t-1$.

La información que disponemos para realizar la segmentación del fotograma $B(t)$ es:

- El fotograma $B(t-1)$.
- La segmentación $\Gamma_{B(t-1)}$ del fotograma $B(t-1)$.
- El descriptor $\alpha_{interes}^{B(t-1)}$ de la región de interés del fotograma $B(t-1)$.
- El fotograma $B(t)$.

El primer paso para el procesado del fotograma $B(t)$ es definir la zona de búsqueda de la región de interés. Para ello usaremos $\Gamma_{B(t-1)}$, que nos indica cuál era la posición del objeto en $B(t-1)$. Dilatamos la región $R_{interes}^{B(t-1)}$ mediante un elemento estructurante lo suficientemente grande para asegurarnos de que abarca la zona donde posiblemente se encontrará el objeto en $B(t)$. Sólo efectuaremos la operación diferencia sobre el área marcada por esta dilatación.

La figura 3.12 muestra la operación realizada. La subfigura (a) es el fotograma $B(t-1)$. En la subfigura (b) se muestra la zona dilatada donde se espera que se encuentre el objeto en el fotograma $B(t)$ representado en la subfigura (c). Como se puede observar, la predicción tiene éxito ya que el movimiento del músculo de un fotograma a otro es bastante débil.

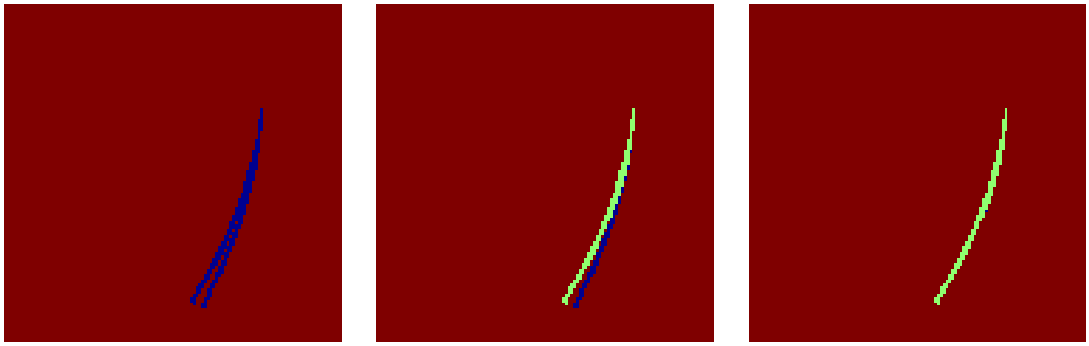


(a) (b) (c)

Figura 3.12.: Predicción de la posición del objeto a partir de la segmentación de un fotograma anterior. (a) Fotograma $B(t-1)$. (b) Predicción de la posición del objeto en el fotograma $B(t)$ usando un elemento estructurante de 11×11 píxeles. La zona roja es el área donde se espera encontrar el músculo. (c) $B(t)$.

Además de reducir el cómputo necesario para la operación diferencia entre los fotogramas $B(t-1)$ y $B(t)$, también eliminamos aquellas diferencias entre ambos fotogramas debidas al ruido o a pequeñas variaciones de la superficie del agua que no nos interesan para el seguimiento del músculo. Según la ecuación 2.36 sólo se tienen en cuenta las diferencias que superan cierto umbral, precisamente para evitar el efecto negativo del ruido y las variaciones insignificantes. Mediante la predicción basada en la información obtenida del fotograma anterior $B(t-1)$ conseguimos que la elección de este umbral no sea tan crítica ya que la mayor parte de estas variaciones indeseadas, simplemente, ya no se tienen en cuenta.

Se calculará ahora la diferencia entre los dos fotogramas tal como se ha explicado antes, obteniendo una segmentación casi completa del fotograma $B(t)$ sin haber dado comienzo todavía al proceso de competición entre regiones (ver figura 3.13(a)).



(a) (b) (c)

Figura 3.13.: Segmentación del fotograma $B(t)$ antes de comenzar la competición de regiones. (a) La zona azul muestra dónde se ha detectado movimiento. (b) Se planta la semilla de la región de interés usando el descriptor del fotograma anterior $\alpha_{interes}^{B(t-1)}$ y ésta crece en primer lugar. (c) Segmentación final $\Gamma_{B(t)}$.

El siguiente paso es encontrar el sitio idóneo para plantar la semilla de la región de interés. Para ello usaremos el conocimiento a priori que tenemos acerca del descriptor de dicha región $\alpha_{interes}^{B(t-1)}$. Dentro de los píxeles que todavía no han sido excluidos por el preprocesado explicado hasta ahora, se encuentra el punto de mayor similitud con el descriptor de la región de interés en el fotograma anterior $B(t-1)$. Se coloca la semilla en este punto y la región comienza a crecer. Este crecimiento se puede observar en la figura 3.13(b). El crecimiento de la región de interés no sigue el algoritmo descrito por RC sino que se da preferencia a su crecimiento respecto a otras. De esta forma se asegura la conectividad entre todos los píxeles que forman el objeto. Cuando la región correspondiente al objeto ha terminado de crecer, se lanza el proceso de segmentación mediante RC si es necesario, es decir, si quedan píxeles de la imagen sin etiquetar.

Los experimentos realizados con las secuencias de músculos artificiales demuestran que el rendimiento del algoritmo mejora notablemente usando esta técnica de preprocesado. En la mayor parte de las secuencias la aplicación no necesita recurrir a la competición de regiones para

conseguir un seguimiento correcto del objeto. Sólo en las situaciones de oclusión del objeto de interés (ver figura 4.3), el algoritmo se ve obligado a realizar el proceso de competición y fusión de regiones ahorrando así mucho tiempo de proceso.

Capítulo 4.

Análisis de resultados y conclusiones

En esta parte final de la memoria, dedicaremos algunas páginas a comentar e ilustrar los resultados obtenidos con nuestro algoritmo en imágenes de diversos tipos. En primer lugar comentaremos los resultados obtenidos con las secuencias de vídeo estereoscópico y posteriormente testearemos la robustez de la segmentación inicial con diversas imágenes reales. Por último apuntaremos las posibles líneas de trabajo futuro.

4.1. Secuencias de vídeo estereoscópico

A lo largo de la descripción de la aplicación hemos visto varios ejemplos de segmentación de las secuencias estéreo de músculos artificiales. Los resultados obtenidos pueden calificarse de satisfactorios ya que las segmentaciones obtenidas representan fielmente el contorno del músculo real y son aptas para la extracción de los parámetros característicos de éste. Diferentes secuencias con diferentes tipos de músculo en las que varía la silueta de éste y ligeramente su iluminación han servido para probar la eficacia del algoritmo, obteniéndose buenos resultados en todas ellas.

A continuación mostramos la segmentación obtenida para diferentes secuencias de vídeo con diferentes músculos. Todos los experimentos han sido realizados con imágenes de 255 niveles de grises de diferentes tamaños. El equipo informático utilizado es un procesador Pentium 4 con una velocidad de 2.4 GHz y 512 Mb de memoria RAM.

La figura 4.1 muestra la segmentación del fragmento inicial de un vídeo de 60 pares de fotogramas de tamaño 150×150 píxeles (la imagen no está a escala). El músculo ha sido cargado con un peso adicional para comprobar su capacidad de esfuerzo (anilla metálica fijada a su extremo inferior). Observamos que la segmentación en los planos perpendiculares difiere ligeramente a causa del apéndice añadido al músculo. El algoritmo es capaz de segmentar exclusivamente la

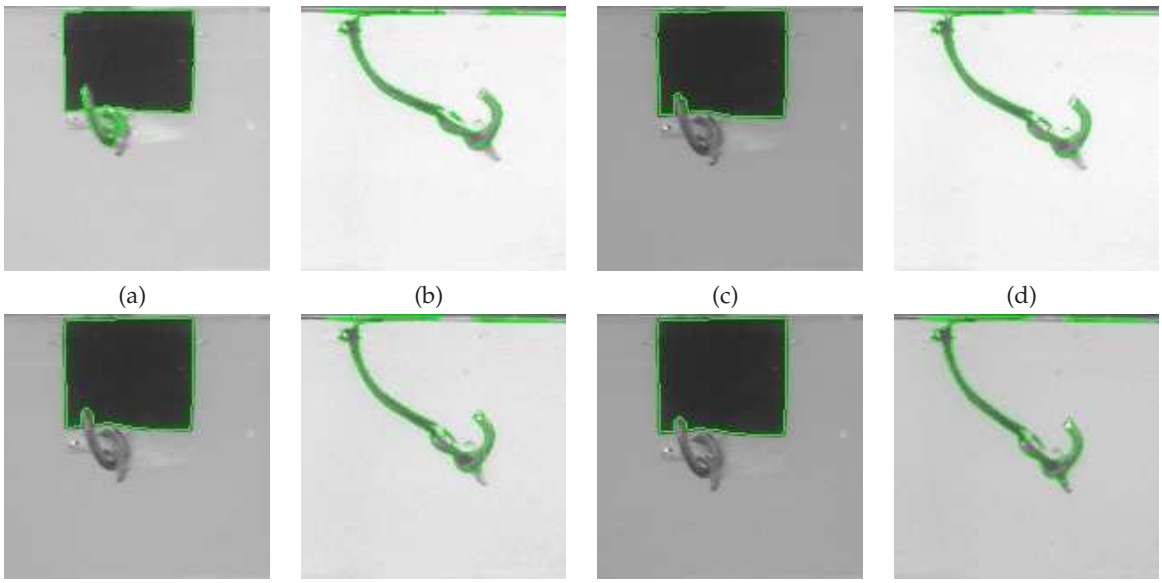
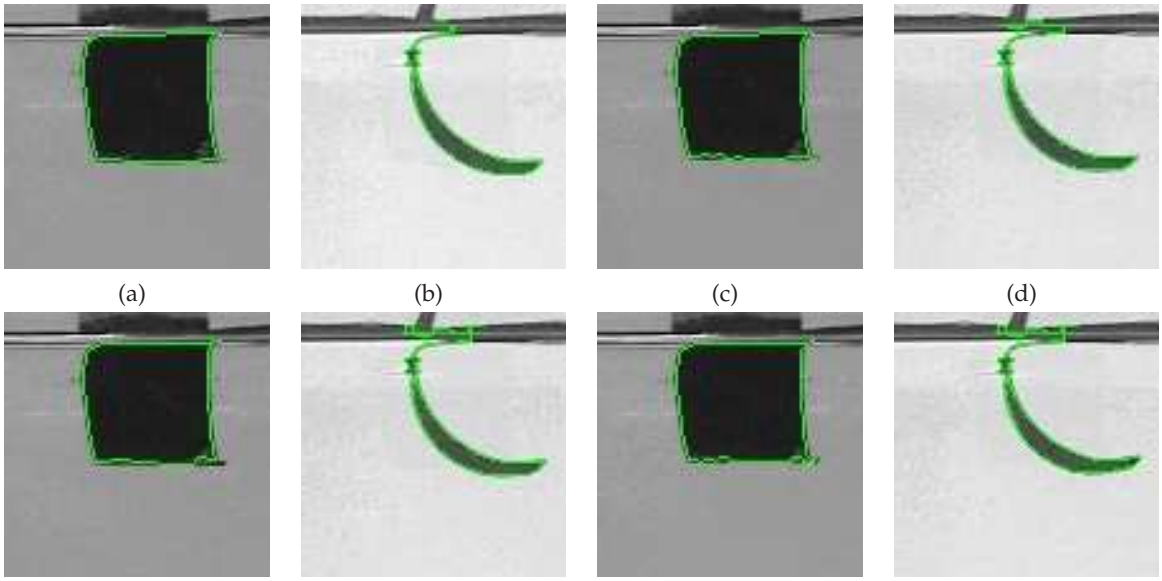


Figura 4.1.: Segmentación de vídeo estereoscópico. (a) y (b) Fotogramas iniciales $A(0)$ y $B(0)$, en sentido lexicográfico los fotogramas subsiguientes.

parte del músculo en el plano XZ mientras que en el plano YZ no discrimina la anilla metálica del contorno del músculo. Para conseguir que el contorno del músculo segmentado sea igual en ambos planos es necesario el uso de algún modelo topológico que, por ejemplo, fije la longitud del contorno y la mantenga constante a lo largo de todo el recorrido. Esta característica no es muy complicada de implementar y se deja como posible mejora en futuras versiones de la aplicación. No obstante la segmentación actual es válida para la extracción de los parámetros del músculo si la mencionada restricción sobre la longitud del músculo se tiene en cuenta en la fase de extracción de parámetros posterior a la fase de segmentación.

La secuencia de la figura 4.2 es el fragmento inicial de un vídeo de 96 pares de fotogramas de tamaño 105×105 píxeles (las imágenes no están a escala). Nótese en este ejemplo y el anterior la variación en la iluminación entre los fotogramas y la presencia de ruido en la imagen. En esta secuencia, la superficie del agua no coincide con el borde de la imagen, esto supone la presencia de más zonas en la imagen o, dicho de otra forma, más carga para el algoritmo. No obstante, los experimentos realizados con las secuencias completas consiguen terminar la segmentación sin agotar los recursos de la máquina ni comprometer la estabilidad del sistema.

En secuencias como la de la figura 4.2 donde la superficie del agua aparece en la imagen se presenta un problema adicional para el algoritmo. Algunos músculos son capaces de curvarse de forma que una parte de ellos emerge por encima de esta superficie. Esta situación provoca la *oclusión* momentánea de parte del músculo por la superficie del agua que presenta, generalmente, una discontinuidad abrupta. La propuesta para abordar este problema ya ha sido comentada



(a) (b) (c) (d)

(e) (f) (g) (h)

Figura 4.2.: Segmentación de vídeo estereoscópico. (a) y (b) Fotogramas iniciales $A(t)$ y $B(t)$, en sentido lexicográfico los fotogramas subsiguientes.

antes: si se desea que la longitud del contorno del músculo se mantenga, es posible añadir esta característica. La implementación actual maneja las situaciones de oclusión como muestra la figura 4.3. En este ejemplo el músculo está en movimiento ascendente y su extremo supera la superficie del agua. La extracción de parámetros a partir de esta secuencia sigue siendo posible gracias a los fotogramas $B(t)$ del plano YZ. Puede observarse que cuando el extremo del músculo emerge de la superficie del agua, el algoritmo lo detecta y continúa con su seguimiento (figura 4.3(n)). En el plano XZ no ocurre lo mismo ya que el músculo, visto desde esta perspectiva, está superpuesto con sigo mismo y el algoritmo no puede (y, en realidad, no debe) distinguir el momento en que éste supera la superficie del agua. Como ya hemos comentado, la extracción de los parámetros del músculo en esta secuencia sigue siendo posible con la actual implementación gracias a la información contenida en la segmentación de los fotogramas del plano YZ.

En vista de los resultados obtenidos, podemos concluir que la aplicación dedicada para la segmentación de videos estereoscópicos de músculos artificiales basada en la competición de regiones es capaz de segmentar correctamente este tipo de fotogramas, detectando el objeto de interés en todos los casos. La intervención del usuario para seleccionar la región correspondiente al músculo no es necesaria si se dispone de la información necesaria a priori. El factor crítico que determina el funcionamiento correcto del algoritmo es la elección de los umbrales. Afortunadamente, los fotogramas captados por el sistema de adquisición no exigen el uso de descriptores de mayor orden que la media y además, las variaciones de éstos en el tiempo no son bruscas, lo que facilita el seguimiento del objeto. La disponibilidad de imágenes estereoscópicas captadas en planos perpendiculares se ha usado para acelerar la segmentación inicial que es la más complicada.

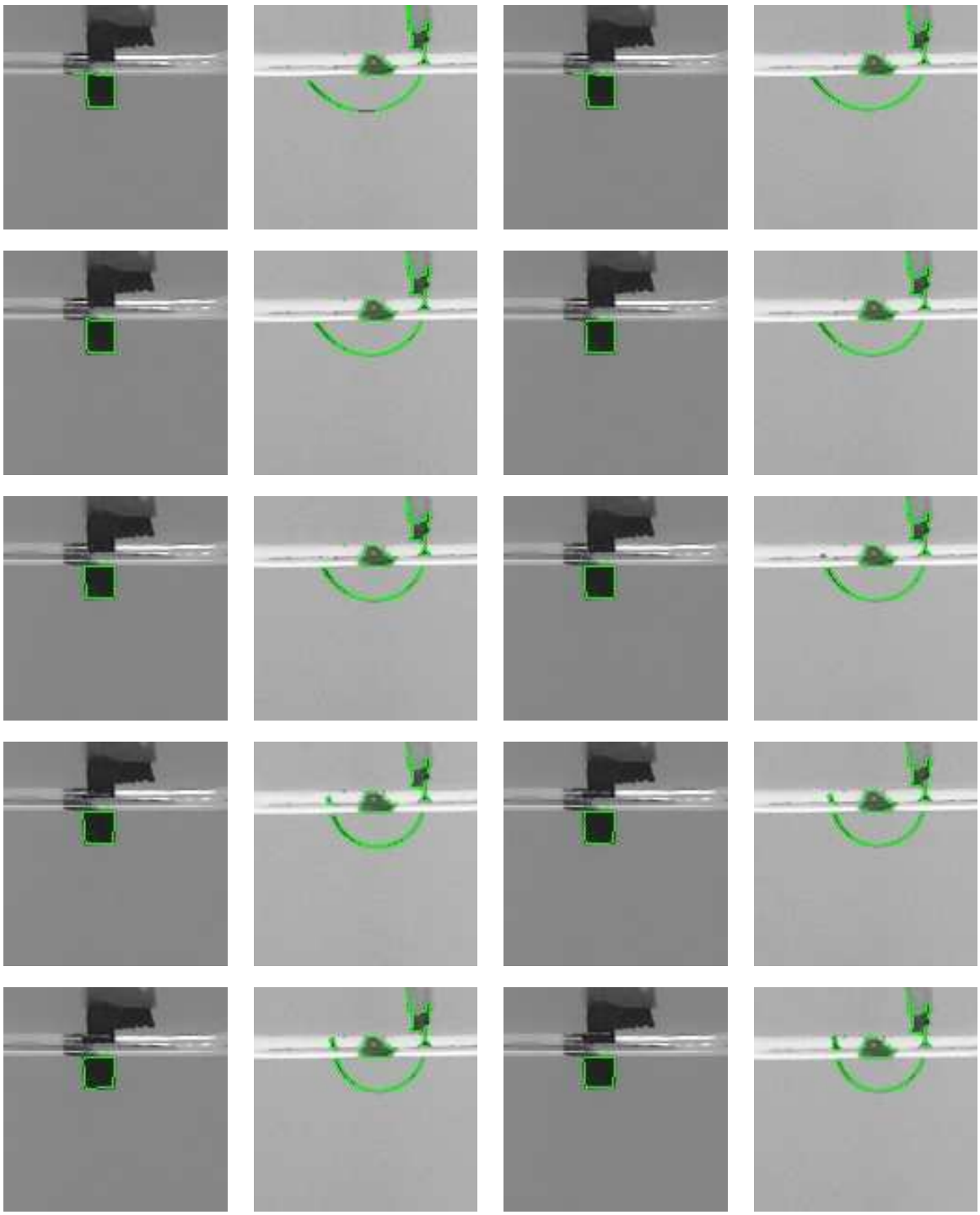


Figura 4.3.: Problema de oclusión en la superficie del agua. Los dos primeros fotogramas son los iniciales $A(0)$ y $B(0)$, en sentido lexicográfico los fotogramas subsiguientes. Nótese que el algoritmo detecta el músculo emergente en los fotogramas del plano YZ ($B(t)$) pero no en los del plano XZ ($A(t)$).

El seguimiento del músculo se ha realizado mediante técnicas de segmentación de vídeo basadas en la diferencia de imágenes combinadas con la aplicación de operadores morfológicos para mejorar la eficiencia del algoritmo. Podemos decir, en vista de lo anterior, que se ha conseguido el objetivo propuesto al inicio de este proyecto al lograr la implementación de un algoritmo eficaz, robusto y eficiente en la tarea de la segmentación y seguimiento de músculos artificiales para su posterior caracterización.

4.2. Experimentos con imágenes de niveles de grises

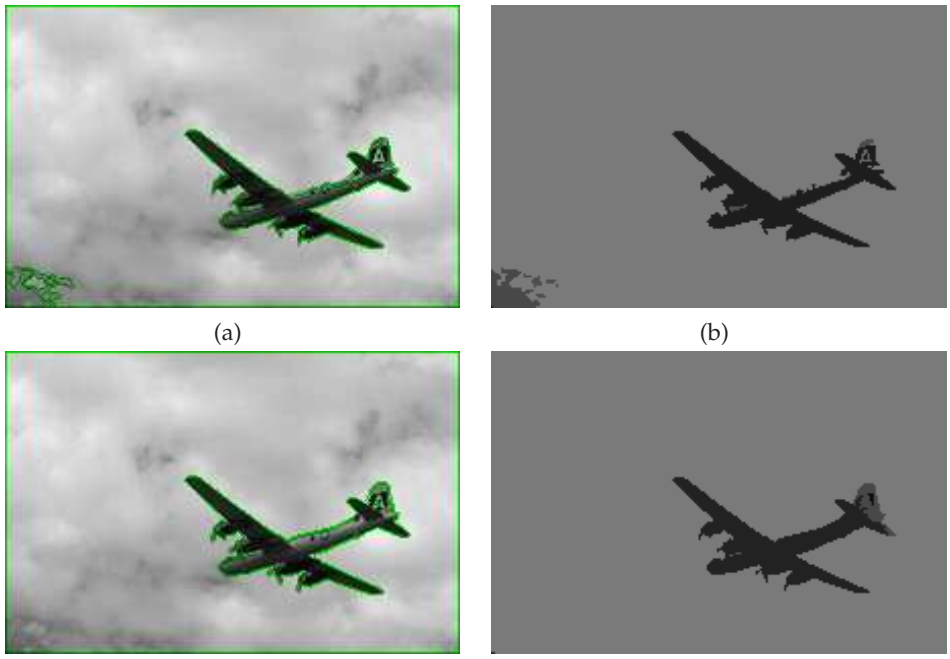
Tras haber evaluado los resultados obtenidos en la segmentación de vídeo estereoscópico, en esta sección analizaremos la eficacia del algoritmo frente a imágenes reales de diferente complejidad. La segmentación de estas imágenes presenta un nuevo reto a nuestra aplicación ya que, aunque los pasos del proceso son, en esencia, los mismos que explicamos en la sección 3.2.3, los datos a procesar son muy diferentes. Se trata, en este caso, de imágenes con mucha más dinámica y riqueza de texturas que obligan al algoritmo a manejar un número mucho mayor de descriptores que en el caso de los músculos artificiales, donde la segmentación final producía, invariablemente, dos únicas regiones finales. Además, no se dispone de ninguna información a priori sobre los datos de las imágenes, por lo que los valores de los umbrales han de ser ajustados por el usuario mediante el método de prueba y error hasta alcanzar resultados razonablemente buenos. Como veremos en los siguientes ejemplos, nuestro algoritmo es capaz de realizar segmentaciones más que aceptables de imágenes bastante complejas, obteniéndose resultados más o menos satisfactorios en función de la naturaleza de los datos y el "afinado" de los umbrales. El equipo informático para la segmentación de las imágenes es el mismo que se usó para las secuencias de la sección anterior. La inicialización del algoritmo se ha llevado a cabo mediante 12 semillas distribuidas aleatoriamente por el dominio de la imagen. El valor de los umbrales de crecimiento y fusión de regiones (T_c y T_f en adelante) se ha ajustado por el método de prueba y error atendiendo a los histogramas de las imágenes y al nivel de segmentación deseado en cada caso.

La primera imagen que servirá para ilustrar el comportamiento del algoritmo frente a imágenes reales es la de la figura 4.4. La figura 4.4(a) es la imagen original de entrada al algoritmo, en este caso, un avión bajo un cielo nublado. Las imágenes de entrada serán, en todos los ejemplos, imágenes RGB o de "color verdadero" aunque el algoritmo, como ya hemos comentado anteriormente, tratará con la imagen en escala de grises. En la figura 4.4(b) se muestra la imagen de entrada a 256 niveles de grises con el resultado de la segmentación superpuesto en color verde. La figura 4.4(c) muestra el resultado de la segmentación representando todas las regiones generadas por el algoritmo. El nivel de intensidad de gris que representa cada región corresponde al valor de su descriptor al final de la ejecución del algoritmo. La segmentación obtenida para la figura 4.4 se ha logrado mediante unos umbrales de crecimiento (T_c) y fusión de regiones (T_f) de 40 y 20.

(a) (b) (c)

Figura 4.4.: Resultados de la segmentación con umbrales $T_c = 40$ y $T_f = 20$. (a) Imagen original. (b) Imagen a 256 niveles de grises con la segmentación final en verde. (c) Representación de los descriptores regionales.

Para ilustrar cómo se puede controlar el nivel de segmentación del algoritmo mediante el valor de los umbrales incluimos la figura 4.5. En una segunda segmentación, se han usado unos umbrales de $T_c = 50$ y $T_f = 30$. Nótese que en la segmentación de la figura 4.5(b) no aparecen los bordes de las nubes en la esquina inferior izquierda de la imagen, lo que puede interpretarse como positivo si lo que nos interesa es únicamente la silueta del avión. No obstante, en esta segunda segmentación no aparece tan bien definida la letra "A" de la cola del avión, lo que, en ciertas aplicaciones, podría considerarse como un fallo del algoritmo. Este compromiso en la elección de los umbrales es un problema que se repite en todos los ejemplos considerados en esta sección, y, en general, en todas las aplicaciones de segmentación.



(a) (b)

(c) (d)

Figura 4.5.: Segmentación con diferentes umbrales. (a) Bordes generados por la segmentación con umbrales $T_c = 40$ y $T_f = 20$. (b) Regiones generadas con umbrales $T_c = 40$ y $T_f = 20$. (c) y (d) muestran el resultado con umbrales $T_c = 50$ y $T_f = 30$.

En el siguiente ejemplo (fig. 4.6) podemos observar una de las principales limitaciones de la actual implementación del algoritmo. El uso de un único descriptor estadístico no es suficiente para encontrar los verdaderos bordes objetivo en aquellas fronteras donde la intensidad de los píxeles es similar. La figura 4.6(a) muestra la imagen en color de un caballo y un potro en una pradera. En la imagen RGB original se distinguen claramente las siluetas de los animales sobre el campo pero en la imagen a niveles de grises no ocurre lo mismo. Los píxeles de la grupa de los caballos tienen un nivel muy parecido a los píxeles de la pradera y el algoritmo es incapaz de segmentar correctamente la totalidad de la silueta de ambos animales. El uso de umbrales más restrictivos no soluciona este problema y, además, provoca la sobresegmentación de otras zonas de la imagen.

(a) (b) (c)

Figura 4.6.: Resultados de la segmentación con umbrales $T_c = 50$ y $T_f = 30$.

Para completar esta sección de experimentos con imágenes reales, incluimos a continuación (figura 4.7) los resultados obtenidos por nuestro algoritmo sobre una serie de imágenes de diferente naturaleza.

Los ejemplos vistos en estas figuras muestran la capacidad de nuestro algoritmo para la segmentación de imágenes reales, mucho más complejas que las de músculos artificiales. La actual implementación, con un único descriptor regional, resulta válida para la segmentación de imágenes con un buen contraste, donde el objetivo está bien diferenciado del fondo. El nivel de segmentación deseado puede ajustarse modificando los valores de los umbrales de crecimiento y fusión T_c y T_f. En aquellas imágenes que presentan niveles de intensidad de gris similares para el fondo y objeto a segmentar, el algoritmo tiene problemas para detectar los bordes con precisión. Esto sugiere un aumento del número de descriptores (varianza, texturas...) que maneja el algoritmo si la aplicación requiere cierto nivel de precisión en imágenes de este tipo. No obstante y, en vista de los resultados obtenidos, podemos concluir que el algoritmo es perfectamente válido para la segmentación de una gran variedad de imágenes reales (ver figura 4.7) donde las condiciones de iluminación no están controladas en absoluto y la información del objetivo a priori es prácticamente nula.

Llegados a esta fase de conclusiones finales, he de añadir que, de entre todas las motivaciones para la realización de este proyecto, la más importante es de índole personal. Siempre me han interesado los procesos de tratamiento de la imagen, desde los sistemas analógicos de televisión

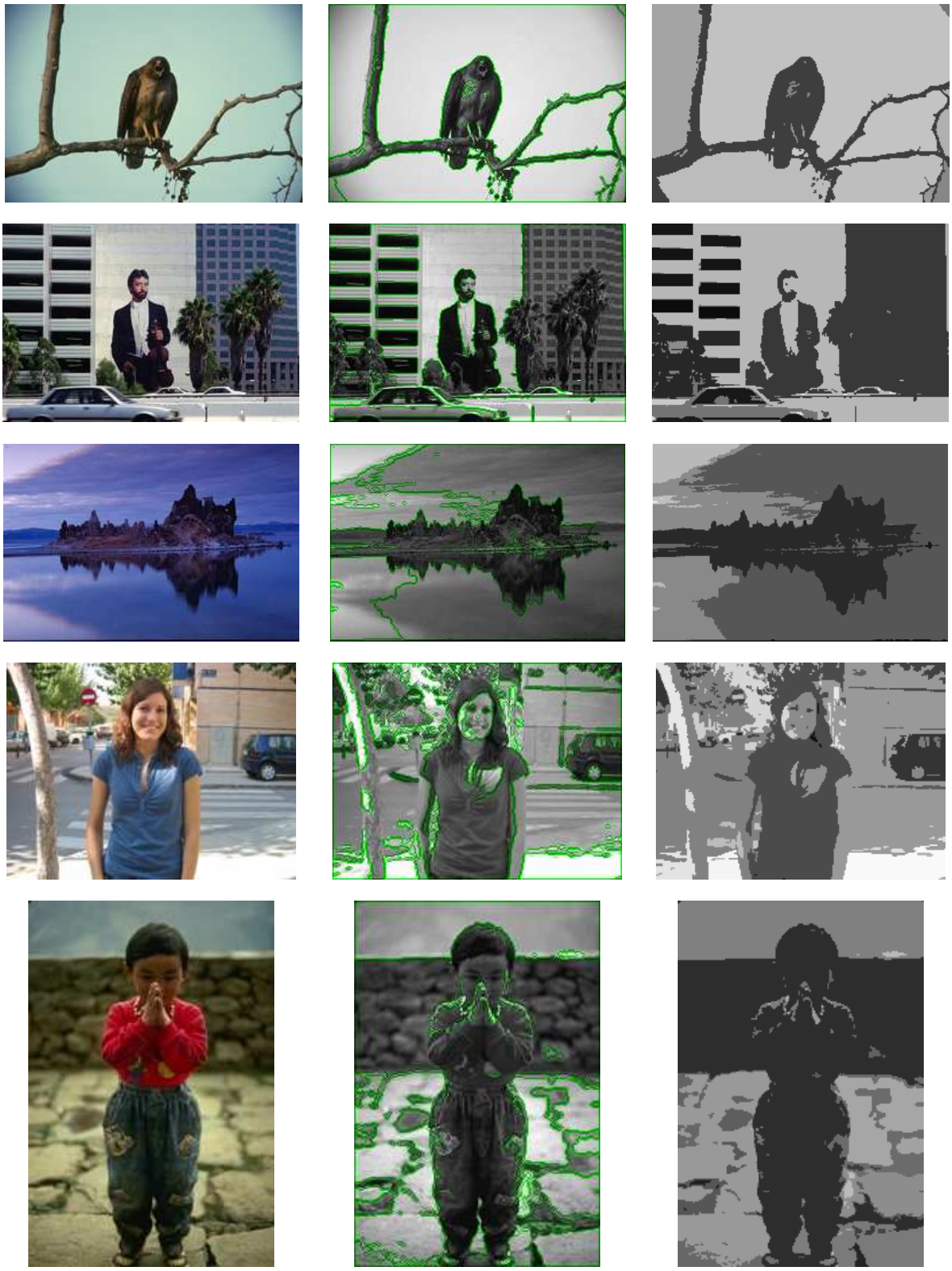


Figura 4.7.: Resultados obtenidos con diversas imágenes tras ajustar los umbrales T_c y T_f en cada caso.

hasta los métodos digitales de compresión y los sistemas de visión artificial. Es una lástima la ausencia de asignaturas relacionadas con estos temas en los programas de formación universitarios. Lo más cercano a éstos que he encontrado a lo largo de mi formación académica han sido un par de asignaturas optativas que sólo trataban conceptos muy básicos, con poca profundidad y de forma estrictamente teórica. La realización de este proyecto me ha permitido descubrir la cantidad de investigaciones relacionadas con la segmentación de imágenes y la visión artificial en general. Es un enorme campo de investigación que todavía está abierto y en el que la creatividad del investigador es, quizá, tan importante como sus conocimientos. Prueba de ello son los múltiples y diferentes enfoques que se plantean para la resolución de un mismo problema. El descubrimiento de este pequeño gran mundo y el conocimiento adquirido a lo largo de la implementación final me ha reportado, quizá, más satisfacciones que la consecución del objetivo concreto. Esta curiosidad por conocer las técnicas usadas para conseguir que una máquina sea capaz de leer la matrícula de un coche, discriminar frutas en mal estado o distinguir entre muestras de tejidos sanos y enfermos ha sido, sin duda, mi principal motivación para la realización de este proyecto.

4.3. Trabajo Futuro

Las líneas de trabajo futuro para la mejora de la actual implementación apuntan hacia la obtención de un contorno de segmentación robusto frente a los problemas de oclusión en el seguimiento del objeto. Como hemos dicho anteriormente, estos problemas pueden solucionarse mediante la adición de un modelo topológico al contorno de la segmentación. Por ejemplo, puede fijarse la longitud del contorno para que éste se mantenga constante durante el desplazamiento del músculo en ambos planos.

El método de extracción de los parámetros de los músculos no se ha tenido en cuenta durante la implementación de la aplicación. La extracción de un modelo en 3D del contorno del músculo puede ser realizada a partir de los datos que proporciona la implementación actual y se propone para nuevas versiones de ésta.

El uso de más descriptores para las regiones, si bien en esta aplicación no ha sido necesario, puede ser útil para la segmentación de imágenes más complejas. Esta mejora consiste solamente en aumentar la dimensionalidad de los datos a manejar y, por tanto, aumentar también el coste computacional. Para conseguir este mismo objetivo, también puede ser útil incluir, entre los criterios de segmentación del algoritmo, algún sistema de detección de bordes que, lógicamente, también añadiría un coste adicional.

Apéndice A.

Inferencia Bayesiana

La mayoría de los algoritmos de segmentación se pueden describir secuencialmente en varias etapas. Tras los pasos de filtrado y extracción de parámetros de interés es imprescindible tomar una decisión en función de la información disponible acerca de la imagen. La segmentación final depende en gran medida del criterio seguido en esta última etapa. El razonamiento bayesiano proporciona un enfoque probabilístico a estas decisiones. Está basado en la suposición de que las cantidades de interés son gobernadas por distribuciones de probabilidad y que se pueden tomar decisiones óptimas razonando sobre estas probabilidades junto con los datos obtenidos. Este enfoque está siendo utilizado en multitud de campos de investigación, entre los que cabe destacar la robótica móvil y la visión por computador, ambas relacionados con el contenido de este proyecto. En este apéndice definimos dos de las herramientas más usadas en el campo de la visión artificial: el teorema de Bayes y el principio de longitud de descripción mínima.

A.1. Teorema de Bayes

A menudo nos surgen problemas en los cuales estamos interesados en determinar la mejor hipótesis h, dados los datos que hemos observado D. Una forma más correcta de expresar esto es decir que buscamos la hipótesis h más probable, dados los datos observados D más un conocimiento inicial sobre las probabilidades a priori de h. El teorema de Bayes nos proporciona un método directo para calcular estas probabilidades.

El teorema de Bayes se define con la siguiente ecuación:

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)} \ , \tag{A.1}$$

donde $P(h)$ es el conocimiento inicial que tenemos acerca de que la hipótesis h sea la correcta. Se le suele denominar la probabilidad a priori de h. $P(D)$ se define de forma similar, pero esta vez sobre

los datos D.$P(D|h)$ denota la probabilidad de observar los datos D dado que tenemos la hipótesis h. Se le suele denominar verosimilitud. Por último, $P(h|D)$ es la probabilidad a posteriori que la hipótesis h tiene, dados los datos observados D. En la mayoría de problemas donde se plantea la inferencia bayesiana, se parte de un conjunto de hipótesis H y se trata de encontrar la hipótesis más probable $h \in H$. De esta forma, a esta hipótesis más probable se le suele denominar hipótesis *maximum a posteriori* o MAP. Utilizando el teorema de Bayes, diremos que h_{MAP} es una hipótesis MAP de acuerdo a:

$$h_{MAP} = \arg \max_{h \in H} P(h|D) = \arg \max_{h \in H} \frac{P(D|h)P(h)}{P(D)} = \arg \max_{h \in H} P(D|h)P(h) \ . \tag{A.2}$$

En el último paso hemos eliminado $P(D)$ porque es independiente de h.

En algunos casos todas las probabilidades en H son igualmente probables a priori ($P(h_i) = P(h_j), \forall h_i, h_j \in H$). En este caso sólo utilizaríamos el término de verosimilitud, $P(D|h)$, y podemos simplificar aún más la anterior ecuación:

$$h_{ML} = \arg \max_{h \in H} P(D|h) \ , \tag{A.3}$$

donde a la hipótesis h_{ML} se le suele nombrar como hipótesis de *máxima verosimilitud* (Maximum Likelihood).

Supongamos ahora que debemos elegir entre dos hipótesis, h_1 y h_2, dados los datos D. El criterio de elección para responder de forma eficiente sería seleccionar la hipótesis más probable. Es decir, aplicaríamos lo que se conoce como regla de decisión:

$$\text{si } P(h_1|D) > P(h_2|D) \text{ elegir } h_1\text{, sino elegir } h_2$$

Si aplicamos la regla de Bayes a cada término y agrupamos nos queda:

$$\frac{P(D|h_1)}{P(D|h_2)} > \frac{P(h_2)}{P(h_1)} \ , \tag{A.4}$$

donde el miembro de la izquierda se conoce como ratio de verosimilitud y el de la derecha como ratio a priori. Aplicando logaritmos a ambos lados de la ecuación:

$$\ln \frac{P(D|h_1)}{P(D|h_2)} > \ln \frac{P(h_2)}{P(h_1)} \ . \tag{A.5}$$

En ausencia de información a priori todas las hipótesis son igualmente probables y el término de la derecha es $\ln 1 = 0$. La regla de decisión en ausencia de información a priori queda:

$$\text{si } \ln \tfrac{P(D|h_1)}{P(D|h_2)} > 0\text{, elegir } h_1\text{, sino elegir } h_2$$

A.2. Principio de longitud de descripción mínima

El principio de longitud de descripción mínima MDL (*minimum description length*) puede ser resumido como "'elegir la explicación más corta a los datos observados"'. Esta íntimamente relacionado con el criterio MAP antes comentado, incorporando conceptos básicos de teoría de la información. Retomando la definición de h_{MAP} de la ecuación A.2 y expresando esta ecuación en términos de la maximización de log_2:

$$h_{MAP} = \log_2 \arg\max_{h \in H} P(D|h) + \log_2 P(h) \ . \tag{A.6}$$

O, alternativamente, minimizando el negativo de esta cantidad

$$h_{MAP} = -\log_2 \arg\min_{h \in H} P(D|h) - \log_2 P(h) \ . \tag{A.7}$$

Esta última ecuación puede ser interpretada como que se prefieren hipótesis cortas. Cada uno de estos términos se puede entender como la longitud de descripción de las distribuciones bajo una codificación óptima. No vamos a entrar en comentar los términos de teoría de información. El principio MDL recomienda la elección de las hipótesis que minimizan estas dos longitudes de descripción. Así, este principio se puede definir como elegir la hipótesis h_{MDL} dada:

$$h_{MDL} = \arg\max_{h \in H} L_{C_1} P(D|h) + L_{C_2} P(h) \ , \tag{A.8}$$

siendo L_{C_i} la longitud de descripción del mensaje i con respecto a C, que es el número de bits requeridos para codificar el mensaje i utilizando el código C. En el caso de que C_2 sea la codificación óptima de las hipótesis h y C_1 sea la codificación óptima de $(D|h)$, entonces $h_{MDL} = h_{MAP}$.

A.3. Hipótesis de máxima verosimilitud y error cuadrático medio

También vamos a demostrar una equivalencia entre la hipótesis de máxima verosimilitud y el método que encuentra la hipótesis que minimiza el error cuadrático medio. Teniendo en cuenta la ecuación A.3 y asumiendo como independientes los datos $D = (d_1, d_2, \ldots, d_m)$ dado h podemos escribir $P(D|h)$ como el producto de los distintos $P(d_i|h)$:

$$h_{ML} = \arg\max_{h \in H} \prod_{i=1}^{m} P(d_i|h) \ . \tag{A.9}$$

Suponiendo funciones de distribución normales con media μ y varianza σ^2, hacemos coincidir la hipótesis h con la media μ. Sustituyendo en la ecuación A.9 tenemos:

$$h_{ML} = \arg\max_{h \in H} \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp -\frac{1}{2\sigma^2}(d_i - \mu)^2 \ . \tag{A.10}$$

Siendo lnP una función monótona de P, maximizar lnP equivale a maximizar P, y eliminando la varianza, que no depende de la hipótesis h, nos queda:

$$h_{ML} = \arg\max_{h \in H} \sum_{i=1}^{m} -\frac{1}{2\sigma^2}(d_i - \mu)^2 = \arg\min_{h \in H} \sum_{i=1}^{m} (d_i - \mu)^2 \ . \tag{A.11}$$

La ecuación A.11 muestra que la hipótesis de máxima verosimilitud es la que minimiza la suma de los errores cuadráticos entre la hipótesis y los datos, siempre en los supuestos antes mencionados.

Apéndice B.

Morfología Matemática

La Morfología Matemática es una teoría cuyo objetivo es el estudio de estructuras espaciales. Se basa en la teoría de conjuntos y representa una potente técnica de análisis de imagen. La Morfología Matemática empieza a desarrollarse a mediados de los sesenta gracias a las ideas de G. Matheron en relación con el estudio de la geometría de medios porosos.

La Morfología Matemática considera los objetos contenidos en una imagen como *conjuntos*. En una imagen binaria la dualidad es obvia: el conjunto que representa al objeto es el formado por todos los píxeles con valor uno. Las imágenes a niveles de gris también son consideradas como conjuntos a través del *grafo* o el *subgrafo*. El grafo G de una imagen f es el conjunto de puntos (x, t) tal que x pertenece a $\mathbb{Z}^{\nvDash}$ y $t = f(x)$:

$$G(f) = \{(x, t) \in \mathbb{Z}^2 \times N_0 : t = f(x)\} \ . \tag{B.1}$$

El subgrafo, SG, de una imagen, f, es el conjunto de puntos de $\mathbb{Z}^{\nvDash} \times N_0$ que están por debajo del grafo y por encima del plano donde está definida la imagen:

$$GS(f) = \{(x, t) \in \mathbb{Z}^2 \times N_0 : 0 \leq t \leq f(x)\} \ . \tag{B.2}$$

Los operadores morfológicos extraen información sobre las estructuras relevantes presentes en la imagen comparándolas con una serie de conjuntos con formas predeterminadas a los que se denomina *elementos estructurantes* (SE). La forma y el tamaño de un elemento estructurante debe elegirse en función de las propiedades geométricas de los objetos representados en la imagen sobre los que queremos información. La figura B.1 muestra un SE cuadrado de 3×3 píxeles. Más adelante veremos que el elemento estructurante actúa como una sonda que analiza el entorno del píxel.

Antes de introducir los operadores morfológicos básicos se van a presentar los operadores de conjuntos en que estos se basan a la vez que es establecida la notación.

1	1	1
1	1	1
1	1	1

Figura B.1.: Elemento estructurante cuadrado de 3×3 píxeles.

B.1. Operadores de conjuntos

Los primeros operadores de conjuntos a que nos referiremos son la unión, $\cup$, y la intersección, $\cap$. A nivel funcional, la unión se traduce en el máximo punto a punto, $\vee$, y la intersección en el mínimo punto a punto, $\wedge$, de las imágenes a niveles de gris dadas por las funciones f y g:

$$(f \vee g)(x) = \text{máx}[f(x), g(x)] \; ; \tag{B.3}$$

$$(f \wedge g)(x) = \text{mín}[f(x), g(x)] \; . \tag{B.4}$$

A nivel conjuntista, estas operaciones pueden definirse en términos de uniones e intersecciones de subgrafos:

$$SG(f \vee g) = SG(f) \cup SG(g) \; ; \tag{B.5}$$

$$SG(f \wedge g) = SG(f) \cap SG(g) \; . \tag{B.6}$$

La *complementación* es otro operador básico. El complemento de una imagen f, denotado por f^c, se define en cada píxel x como la diferencia entre t_{max} y el valor $f(x)$:

$$f^c(x) = t_{max} - f(x) \; . \tag{B.7}$$

La *traslación* de una imagen, f, por un vector, b, se denota por f_b. El valor de la imagen trasladada en un píxel x es igual al valor de la imagen original en la posición $x - b$:

$$f_b(x) = f(x - b) \; . \tag{B.8}$$

Todos los operadores morfológicos se basan en la combinación de los operadores intersección (mínimo punto a punto), unión (máximo punto a punto), complementación y traslación.

B.2. Operadores morfológicos básicos

B.2.1. Erosión y dilatación

La *erosión* de un conjunto X por un elemento estructurante B es denotado por $\epsilon_B(X)$ y se define como el conjunto de puntos, x, tal que la versión trasladada de B por x, B_x, está contenida en X:

$$\epsilon_B(X) = \{x : B_x \subseteq X\} \ . \tag{B.9}$$

Otra posible definición del mismo concepto viene dada por la siguiente ecuación:

$$\epsilon_B(X) = \bigcap_{b \in B} X_{-b} \ . \tag{B.10}$$

Esta última definición conjuntista puede ser extendida directamente al caso funcional: la erosión de una imagen f por un elemento estructurante B es denotado por $\epsilon_B(f)$ y se define como el mínimo de las traslaciones de f por los vectores $-b$ de B:

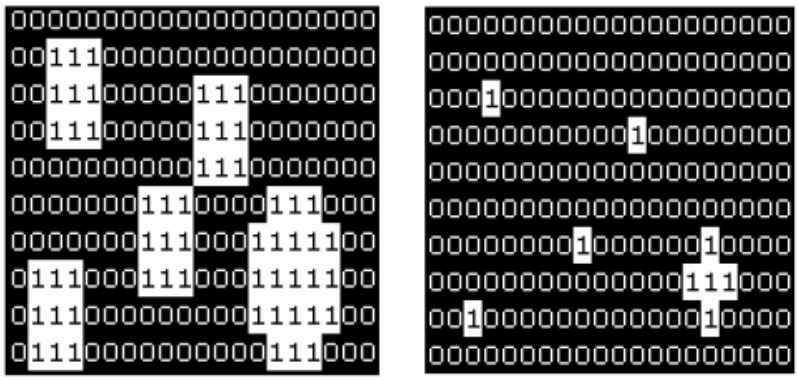
$$\epsilon_B(f) = \bigwedge_{b \in B} f_{-b} \ , \tag{B.11}$$

de forma que el valor de la erosión en un píxel x no es otra cosa que el mínimo de la imagen en la ventana definida por el elemento estructurante trasladado por x:

$$[\epsilon_B(f)](x) = \min_{b \in B} f(x + b) \ . \tag{B.12}$$

Los efectos de la erosión sobre una imagen pueden verse en la figura B.2. Obviamente, el resultado depende en gran medida del elemento estructurante que estemos usando. En este caso hemos utilizado un SE cuadrado de 3×3 píxeles como el de la figura B.1. En este caso, un píxel tomará el valor 1 en la imagen procesada, *si ese mismo píxel y todos sus vecinos valían 1* en la imagen original. Los efectos típicos de la erosión sobre una imagen son los siguientes:

- Hacer más finas las figuras.
- Si los objetos son pequeños pueden desaparecer.
- Si los objetos tienen istmos pequeños pueden partirse.
- Los salientes estrechos son eliminados.
- Si se aplica reiteradamente elimina todos los píxeles a 1 de la imagen.



(a) (b)

Figura B.2.: Ejemplo de erosión de una imagen binaria. (a) Imagen original. (b) Imagen tras la erosión con un elemento estructurante 3×3.

La *dilatación* es el operador dual de la erosión. La dilatación de un conjunto X por un elemento estructurante B se denota $\delta_B(X)$ y se define como el conjunto de puntos x tal que B_x intersecta con X:

$$\delta_B(X) = \{x : B_x \cap X \neq \emptyset\} \ . \tag{B.13}$$

La anterior ecuación puede ser reescrita como la unión de conjuntos trasladados mediante puntos del elemento estructurante:

$$\delta_B(X) = \bigcup_{b \in B} X_{-b} \ . \tag{B.14}$$

La extensión del concepto a imágenes binarias y a niveles de gris es directa: la dilatación de una imagen a niveles de gris, f, mediante un elemento estructurante B se denota por $\delta_B(f)$ y se define como el máximo de la traslación de f mediante los vectores $-b$ de B:

$$\delta_B(f) = \bigvee_{b \in B} f_{-b} \ . \tag{B.15}$$

Dicho de otra manera, el valor dilatado en un píxel x es el valor máximo de la imagen en la ventana definida por el elemento estructurante trasladado por x:

$$[\delta_B(f)](x) = \max_{b \in B} f_{x+b} \ . \tag{B.16}$$

La figura B.3 ilustra el efecto de la dilatación de una imagen binaria con un elemento estructurante como el de la figura B.1. En este ejemplo, un píxel tomará el valor 1 en la imagen procesada, *si ese mismo píxel o cualquiera de sus vecinos valían 1* en la imagen original.

Los efectos típicos de la dilatación de una imagen son los siguientes:

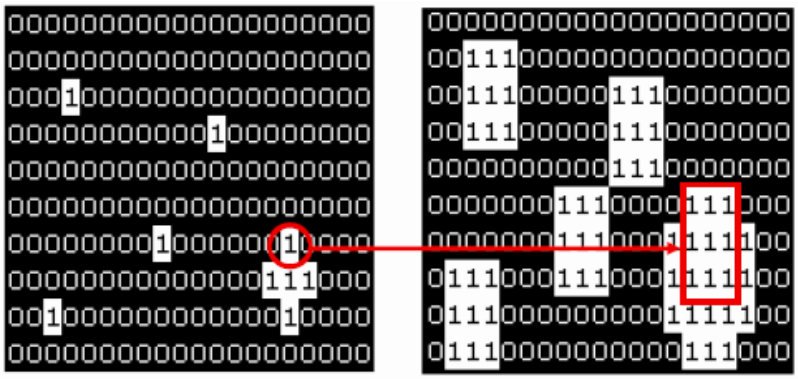


Figura B.3.: Ejemplo de dilatación de una imagen binaria con un elemento estructurante 3×3.

- Hacer más gruesas las figuras.
- Puede unir objetos si están próximos.
- Reduce/elimina agujeros.
- Los entrantes estrechos son rellenados.

Por último ilustraremos el efecto de las erosiones y dilataciones sobre imágenes binarias con la figura B.4. En cada uno de los cuadrantes se ha usado un elemento estructurante diferente de forma que se puede apreciar el diferente efecto que provoca la misma operación para diferentes SE.

B.2.2. Aperturas y cierres

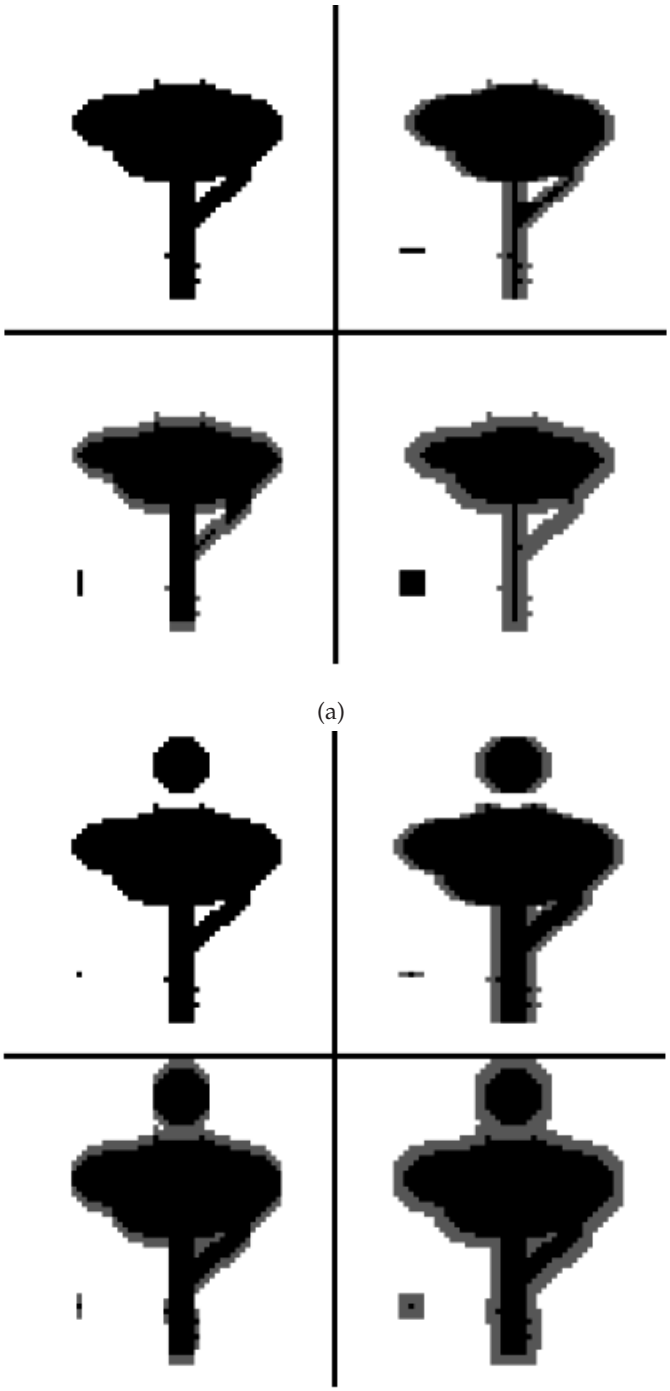
La *apertura* γ de una imagen f por un elemento estructurante B se denota por $\gamma_B(f)$ y se define como la erosión de f por B seguida de la dilatación por $\breve{B}$, donde $\breve{B} = \{-b : b \in B\}$:

$$\gamma_B(f) = \delta_{\breve{B}}[\epsilon_B(f)] \; . \tag{B.17}$$

Cuando partimos de un conjunto X, su apertura es la unión de todas las versiones del elemento estructurante trasladadas que caben dentro de X:

$$\gamma_B(f) = \bigcup \{B_x : B_x \subseteq X, x \in \mathbb{R}^2\} \; . \tag{B.18}$$

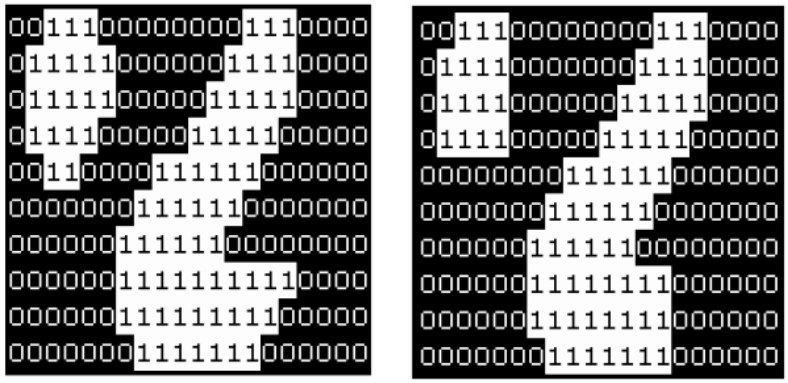
En la figura B.5 podemos ver el efecto de la apertura de una imagen binaria que no es otra cosa que la erosión y dilatación (por este orden) de la imagen original. Como se puede ver, la



(a)

(b)

Figura B.4.: Erosión y dilatación de imágenes binarias con diferentes elementos estructurantes. (a) Erosión. (b) Dilatación.



(a) (b)

Figura B.5.: Ejemplo de apertura de una imagen binaria. (a) Imagen original. (b) Imagen tras la apertura.

apertura elimina los salientes estrechos, así como los objetos más pequeños que el elemento estructurante. Además, la operación de apertura es idempotente.

El *cierre* de una imagen f mediante un elemento estructurante B se denota por $\phi_B(f)$ y se define como la dilatación de f mediante un elemento estructurante B seguido por la erosión con el elemento estructurante traspuesto $\breve{B}$:

$$\phi_B(f) = \epsilon_{\breve{B}}[\delta_B(f)] \ . \tag{B.19}$$

El cierre del conjunto X se define como:

$$\phi_B(f) = \left[\bigcup\{B_x : B_x \subseteq X^c, x \in \mathbb{R}^2\}\right]^c \ , \tag{B.20}$$

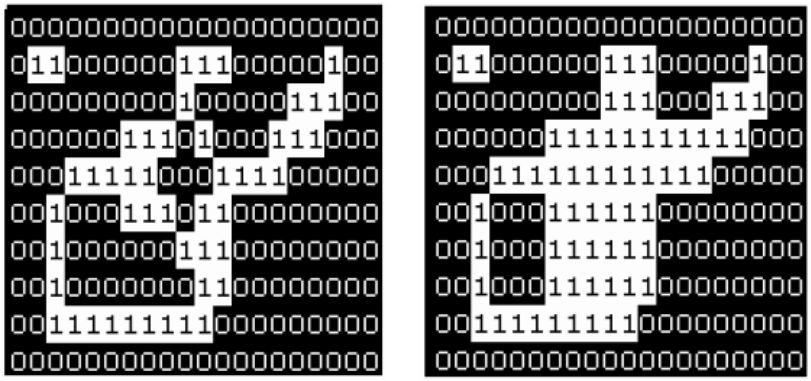
o de forma equivalente:

$$\phi_B(f) = \bigcap\{B_x^c : X \subseteq B_x^c, x \in \mathbb{R}^2\} \ . \tag{B.21}$$

La figura B.6 ilustra el concepto de cierre de una imagen binaria. El cierre es la dilatación y posterior erosión de la imagen. El cierre de imágenes binarias elimina entrantes estrechos, incluyendo aquellos agujeros en los que no quepa el elemento estructurante. Además, la operación de cierre es idempotente.

Intuitivamente, cuando abrimos un conjunto, lo evaluamos desde dentro eliminando aquellos píxeles del objeto que no pueden ser cubiertos mediante alguna traslación del elemento estructurante. Es decir un conjunto siempre contiene a su apertura. Por el contrario, en el cierre evaluamos el conjunto complementario, añadiendo al conjunto aquellos píxeles del complementario que no están contenidos en su apertura. De esta forma, un conjunto siempre está contenido en su cierre. La anterior dualidad se expresa formalmente de la siguiente manera:

$$\gamma_B(f) = [\phi_B(X^c)]^c \ . \tag{B.22}$$



(a) (b)

Figura B.6.: Ejemplo de cierre de una imagen binaria. (a) Imagen original. (b) Imagen tras el cierre.

B.3. Otros filtros

En las secciones anteriores hemos visto los operadores básicos que definen la operaciones en morfología matemática. No obstante, pueden crearse nuevos filtros a partir de estos operadores básicos. Estos nuevos filtros son, en muchos casos, combinaciones alternativas de aperturas y cierres sobre la imagen. Aunque, en esencia, los filtros morfológicos son filtros de suavizado (paso bajo), pueden obtenerse filtros paso alto por diferencia. Por ejemplo, calculando la erosión de una imagen binaria y restándola a la original pueden obtenerse los bordes de los objetos presentes en ella.

Pueden implementarse operadores más sofisticados para la reconstrucción de imágenes y la recuperación de objetos de interés sobre un fondo. Asimismo, la morfología matemática puede usarse para la imposición de mínimos y máximos regionales sobre una imagen que pueden servir como marcadores para la segmentación de una imagen de la que se tiene cierta información a priori (*watersheds*).

Existen además una serie de operadores para imágenes binarias más sofisticados que sirven para adelgazar o engordar objetos sin partirlos o unirlos. Estos operadores operan iterativamente añadiendo o eliminando capas y verificando en cada iteración si se cumplen las condiciones de que el objeto siga sin partirse o unirse con otros. Algunos de estos operadores son *thinning* (adelgazamiento de objetos), *esqueletos* (adelgazamiento manteniendo la forma básica del objeto) y *thicken* (engrosamiento).

Puede encontrarse más información acerca de estos y otros operadores en [26], [27] y [28].

Bibliografía

[1] S. C. Zhu and A. Yuille. *"Region Competition: Unifying Snakes, Region Growing, and Bayes/MDL for Multi-band Image Segementation". IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 18, No. 9*(1996)

[2] R. Verdú, J. Morales, T.F. Otero and L. Weruaga. *"Mechanical Characterization of Artificial Muscles with Computer Vision". Proc. of SPIE Annual Int. Symposium on Smart Structures and Materials*, (2002)

[3] R. Verdú, R. Berenguer, J. Morales, G. Vazquez, T.F. Otero and L. Weruaga. *´´3D Mechanical Characterization of Artificial Muscles with stereoscopic computer vision and active contours". Proc. of IEEE Int. Conf. on Image Processing*, (2003)

[4] R. Verdú, R. Berenguer, J. Morales, G. Vazquez, T.F. Otero and L. Weruaga. *"Mechanical Characterization of the Life Cycle of Artificial Muscles through Stereoscopic Computer Vision and Active Contours". Proc. of IEEE Int. Conf. on Image Processing*, (2005)

[5] T. F. Otero. *"Patent EP-9200095 and EP-9202628"*. (1992)

[6] T. F. Otero and J. M. Sansiñena. *"Bilayer dimensions and movement of artificial muscles". Bioelectrochem. Bioenergetics*, (1997)

[7] BQU-2001-0477. *"Polipirroles; Electrosíntesis y Caracterización de Propiedades Electroquímicas Mediante el Tratamiento Digital de Imágenes"*, (2001)

[8] R. Berenguer and R. Verdú. *"Sistema de visión por computador para tracking automático y caracterización de objetos en 3D". XX Simposium Nacional de la Unión Científica Internacional de Radio (URSI)*, (2005)

[9] Marr, D., and Hildreth, E. *"Theory of Edge Detection". Proceedings of the Royal Society London 207*, (1980)

[10] Marr, D.*"Vision: A Computational Investigation into the Human Representation and Processing of Visual Information",CA: W H Freeman.*(1982)

[11] Canny, J.*"A Computational Approach to Edge Detection"*,(1986)

[12] F. Meyer and S. Beucher,*"Morphological Segmentation"*,(1998)

[13] M. Kass, A. Witkin and D. Terzopoulos,*"Snakes: Active contour models". Int'l J. Comp. Vis.*(1987)

[14] Ch. Xu and J. Prince,*"Snakes, Shapes and Gradient Vector Flow". IEEE Transactions on image Processing, Vol 7 Nº3*,(1998)

[15] S. Beucher,*"The Watershed Transformation Applied to Image Segmentation". Scanning Microscopy 6:299-314*,(1992)

[16] L. Rodney and G.R. Thoma,*"Indexing of Image Content in Spine X-rays". Proceedings of SPIE Storage and Retrieval for Image and Video Databases VIII, vol. 3972*,(2000)

[17] C. K. Chow, T. Kaneko,*"Boundary Detection of Radiographic Images by a Threshold Method". IFIP Congress: 1530-1535*, (1971)

[18] S. C. Zhu and A. Yuille. *"A Unified Theory for Image Segmentation: Region Competition and its Analysis". Havard Robotics Laboratory Technical Report 95-7*, (1995)

[19] M. Tang and S. Ma *"General Scheme of Region Competition Based on Scale Space". IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol 23, Nº12*, (2001)

[20] X. Qing, Y. Jie, D. Siyi *"Texture Segmentation using LBP embedded Region Competition". Electronic Letters on Computer Vision and Image Analysis 5(1):41-47*, (2005)

[21] P. Salembier, F. Marqués, *"Region-based representations of image and video: Segmentation tools for multimedia services". IEEE Trans. on Circuits and Systems for Video Tech. Vol 9, Nº8*, (1999)

[22] D. Freedman, T. Zhang, *"Active Contours for Tracking Distributions". IEEE Trans. on Image Processing*, (2004)

[23] S.C. Zhu, Y. Wu, D.Mumford, *"Filters, Random Fields and Maximum Entropy (FRAME): Towards a Unified Theory for Texture Modeling". International Journal of Computer Vision, Vol 20, Nº3*, (1996)

[24] S.C.Zhu, A. Yuille, *"FORMS: A Flexible Object Recognition and Modeling System". International Journal of Computer Vision, Vol 20, Nº3*, (1996)

[25] R. C. Gonzalez, R. E. Woods *"Digital Image Processing". Addison-Wesley Publishing Comapny Inc.*, (1992)

[26] J. Serra, *"Image analysis and mathematical morphology". Academic Press London*, (1982)

[27] L. Vincent *"Morphological grayscale reconstruction in image analysis: Applications and efficient algorithms". IEEE Trans. on Image Processing*, (1993)

[28] P. Salembier, J. Serra *"Flat zones filtering, connected operators and filters by reconstruction". IEEE Trans. on Image Processing*, (1995)