Centro de Investigación en Matemáticas A.C.

Segmentación asistida usando bordes, textura y color

Tesis

que para obtener el grado de

Maestro en Ciencias con Especialidad en Computación y Matemáticas Industriales

presenta

Josué Tago Pacheco

Director de Tesis

Dr. Mariano J.J. Rivera Meraz

Guanajuato, Gto. *Diciembre de 2007*

A mi abuelita Ani

Agradecimientos

Antes que a nadie quisiera agradecer a mi asesor el Dr. Mariano Rivera quien definitivamente sin su ayuda y asesoría ilimitada esta tesis no hubiera sido posible y menos llevarse a cabo en tan poco tiempo. Me llevo la grata experiencia de poder haber trabajado con él y haberme embarrado un poco de su creatividad y calidad humana. Le agradezco a mis sinodales, al Dr. Flavio Vigueras y al Dr. Rogelio Hasimoto, por tomarse el tiempo de leer esta tesis y por sus útiles comentarios. A todos mis profesores les agradezco mucho por su dedicación a las clases y el continuo cuestionamiento que me provocaban, además de su disposición total.

A mi familia por siempre estar conmigo y ser mi soporte en la vida. Gracias papá por tu apoyo incondicional, mamá por tu amor que siempre abriga mi ser y Damián por el cariño que nos une. A mi abuelita Ani, a quien además está dedicada esta tesis, por la ternura y amor que tengo la dicha de recibir de tí toda la vida. A mi otra familia, mis mejores amigos Abraham, Iván,Jorge y Sergio que sin su camaradería y apoyo mi vida no sería la misma.

A la *comuna hippie pokarera ravera* que sin duda hicieron mi estancia en Guanajuato mucho más interesante por así decirlo. Gracias Angel y Grace por robarme el dinero en el pokar, Carlos por tu gritos, Gerardo por tus lamentos y Pablo por tus locuras. A pesar de todo no pude encontrar tan buenos amigos como ustedes que serán de por vida, no lo olviden. Siguiendo en Guanajuato quiero agradecerle a Edna por su amistad verdadera. A Abby quien, a pesar de conocerla desde hace poco, ahora forma parte de mi vida.

A mi generación en la maestría pero especialmente a Angulo, Gerardo, Juanita, Oscar y Luz con quienes estreché una gran amistad y nos apoyamos los unos a los otros. Gerardo quien merece mención honorífica por haberme aguantado tanto tiempo viviendo juntos, gracias por todo viejo. A *Gallo Muerto No. 5*, conformado por Ivván, Gerardo, Leonel, Víctor y Yarza, por su amistad, los buenos conciertos y las pizzas después de los ensayos.

Finalmente quisiera agradecer a CONACyT y CONCyTEG por haberme apoyado económicamente para que pudiera llevar a cabo mis estudios de maestría.

Índice general

1. Introducción **1**
- 1.1. Segmentación . . . 1
- 1.2. Clasificación de técnicas de segmentación . . . 3
  - 1.2.1. Segmentación basada en características . . . 3
  - 1.2.2. Segmentación basada en regiones . . . 4
  - 1.2.3. Segmentación basada en bordes . . . 6

2. Segmentación asistida **9**
- 2.1. Propiedades deseadas para técnicas de segmentación asistida . . . 9
- 2.2. Trimapa . . . 11

3. Campos aleatorios Markovianos **13**
- 3.1. Definición . . . 13
- 3.2. Campos aleatorios de Gibbs . . . 14
- 3.3. Estimación Bayesiana . . . 14
- 3.4. Campos Markovianos de probabilidad cuadráticos . . . 15

4. Trabajo propuesto **21**
- 4.1. Segmentación de imágenes con QMPF y trimapas . . . 21
- 4.2. Textura . . . 23
  - 4.2.1. Introducción . . . 23
  - 4.2.2. Una taxonomía de modelos de texturas . . . 24
  - 4.2.3. Segmentación de texturas . . . 29
  - 4.2.4. Segmentación de imágenes con color y textura . . . 30
- 4.3. Histogramas vs GMMs . . . 31
  - 4.3.1. Algoritmo Esperanza-Maximización . . . 31
  - 4.3.2. Algoritmo Esperanza Maximización con Estimador de Mínima Longitud de Descripción . . . 36
  - 4.3.3. Verosimilitud a partir de un GMM . . . 39
- 4.4. Segmentación robusta ante imágenes con textura . . . 42
  - 4.4.1. Selección del tamaño de ventana . . . 44
  - 4.4.2. Alternativa al tamaño de ventana . . . 45

5. Resultados y discusión **51**
5.1. Comparación cuantitativa con respecto al estado del arte 51
5.1.1. Elecciones finales sobre características del algoritmo 51
5.1.2. Tablas comparativas 54
5.2. Segmentación de texturas 58

6. Conclusiones y Trabajo a futuro **63**
6.1. Conclusiones 63
6.2. Trabajo a Futuro 64

Índice de figuras

1.1. Segmentación jerárquica automática 2
1.2. Segmentación basada en características aplicando un umbral 4
1.3. Segmentación basada en regiones por *nested cuts* 5
1.4. Segmentación basada en bordes por *level sets* 7

2.1. Ejemplo de un trimapa 11

4.1. Ejemplo de segmentación asistida por QMPF 23
4.2. Rasgos de Haralick 25
4.3. Ejemplo de segmentación de texturas basada en momentos 27
4.4. Ejemplo de segmentación de texturas integrando procesos basado en regiones y basado en bordes 30
4.5. Verosimilitud a partir de un GMM 40
4.6. Diagrama de flujo sobre el umbral de las verosimilitudes 40
4.7. Verosimilitudes distintas modificando al factor ω 41
4.8. Verosimilitudes distintas modificando al umbral h 41
4.9. Imagen sintética con ruido Gaussiano y las verosimilitudes obtenidas con diferentes tamaños de ventana 45
4.10. Imagen real con textura y las verosimilitudes obtenidas con diferentes tamaños de ventana 46
4.11. Alternativa al tamaño de ventana para la descripción de textura . . . 47
4.12. Diagrama de flujo sobre el filtrado de las distancias 48
4.13. Verosimilitudes a partir del filtrado lineal binomial de las distancias . 49
4.14. Máscara de convolución de una difusión Homogénea 49

5.1. Ejemplo de una imagen del *benchmark* de Lasso 52
5.2. Bordes obtenidos con la Norma de las Derivadas Gaussians 53
5.3. Bordes a partir de los términos de regularización del algoritmo QMPF 53
5.4. Verosimilitudes obtenidas a partir de distintos tamaños de ventanas . 54
5.5. Ejemplo de la segmentación de una imagen del *benchmark* de Lasso . 55
5.6. Segmentación de texturas 59
5.7. Segmentación de texturas con la posición del pixel como dos rasgos extras 60
5.8. Segmentación de texturas filtrando las distancias 61

6.1. Imagen contraejemplo 65

Prefacio

La parte central de esta tesis es la segmentación interactiva de imágenes a partir de la información de color, textura y bordes de la imagen. La segmentación interactiva últimamente ha recibido gran atención debido a que la segmentación automática para imágenes en general sigue siendo un problema sin resolverse y sumamente difícil. La segmentación interactiva proporciona información por parte de un experto que resulta sumamente útil para lograr la segmentación deseada.
La forma en que el usuario interactúa con el proceso de segmentación es en base a una técnica llamada *trimapa* donde un experto etiqueta sobre la imagen tres distintas clases. Estas clases marcan cada pixel como objeto(s) a segmentar, fondo y sin clasificar. Los pixeles que no estén clasificados son aquellos sobre los que actúa el proceso de segmentación. Está técnica no está restringida a la segmentación binaria, de hecho puede extenderse fácilmente para segmentar muchas clases de interés dentro de una misma imagen.
Esta tesis se basa en un trabajo anterior de Rivera *et. al* [50] sobre segmentación interactiva binaria de imágenes donde se presenta una nueva función de segmentación basada en modelos de campos aleatorios Markovianos. La función de energía propuesta puede minimizarse de forma eficiente a través de algoritmos de optimización con restricciones bien conocidos, además de que una buena aproximación inicial acelera la convergencia. Este algoritmo de segmentación calcula la probabilidad de que cada pixel pertenezca a una clase, previamente señalada por el trimapa. El cálculo de este campo de probabilidad se diferencia de otras propuestas del estado del arte, basadas en métodos *graph cut*, porque estas encuentran un campo binario.
El principal interés de esta tesis era el de obtener mayor información la imagen para obtener una mejor aproximación inicial, esto es muy importante debido a que por la estructura de la función de segmentación a través de cualquier algoritmo de optimización se asegura la convergencia a un mínimo local. Entonces una mejor aproximación inicial nos puede acercar al mínimo que logre la segmentación deseada y evitar los mínimos incorrectos.
El cálculo de las verosimilitudes de los pixeles no clasificados con respecto a cada clase es el paso previo a la segmentación. En [50] se estiman las distribuciones empíricas de las clases objeto y fondo mediante técnicas de histogramas de las regiones marcadas en el trimapa sin embargo se propuso encontrar esa distribución de forma no paramétrica a través del algoritmo EM MDL que nos permite aproximar la distribución con un modelo de mezcla de Gaussianas. Este cambio además de reportarse como

más adecuado [24], facilitaría la inclusión de mayores características de la imagen. La textura fue la característica de la imagen que queríamos incluir desde un principio, lamentablemente resultaba sumamente dificil poder elegir de que forma se incluiría para mejorar la segmentación. Lo anterior se debe a que para obtener la textura de una imagen existen muchos métodos que funcionan de forma adecuada ante ciertos escenarios pero no existe un método que pueda clasificar cualquier tipo de textura. La propuesta presentada en este trabajo es novedosa y permite lograr una segmentacion robusta ante la textura.
Para probar los cambios realizados se aplicó el algoritmo propuesto sobre un grupo de imágenes de un *benchmark*, sobre el cual se llevó a cabo el aprendizaje de los parámetros y se realizó una comparación cuantitativa con respecto al estado del arte. También se probó la segmentación con imágenes mosaico de texturas.
A continuación se da una breve descripción de cada capítulo de la tesis.

Capítulo 1 - Introducción: Se describe la segmentación de imágenes, el papel que juega en la comunidad de visión computacional y la importancia que recientemente ha adquirido para muchas otras áreas de investigación como la medicina. Posteriormente se presenta la clasificación de las técnicas segmentación donde se describe brevemente cada técnica, sus ventajas y sus desventajas.

Capítulo 2 - Segmentación asistida: En esta sección se habla sobre la alternativa que implica la segmentación asistida para facilitar el problema de segmentar imágenes. Se habla de las propiedades deseadas en la segmentación asistida y sobre los trimapas que son la herramienta que se utiliza en este trabajo para que el usuario interactúe en el proceso de segmentación.

Capítulo 3 - Campos aleatorios Markovianos: Se define lo que es un campo aleatorio Markoviano, un campo aleatorio de Gibbs y se explica la estimación Bayesiana que es la forma en que se plantearon los campos markovianos de probabilidad cuadráticos (QMPF). Posteriormente se dá una clara explicación de como se plantearon los QMPF para resolver el problema de segmentación de imágenes.

Capítulo 4 - Trabajo propuesto: En este capítulo primero se describe la implementación del QMPF utilizando trimapas y el uso de histogramas. Para después explicar cuales fueron los cambios propuestos con el fin de mejorar la segmentación apoyados en el uso de mayor información de la imagen y robustez ante la textura. Se da una explicación de porque en lugar del uso de histogramas se buscó modelación con una mezcla de Gaussians mediante el algoritmo EM (por sus siglas en inglés) con estimador de mínima longitud de descripción (el cual se explica de forma detallada). También se introduce el concepto de textura, su taxonomía y algunos procesos de segmentar texturas, para posteriormente explicar la forma que se lidió para incluir la información que las texturas proporciona. Se propone un algoritmo basado en estimacion de máxima verosimilitud que incluye infomación del contexto espacial de cada pixel. El algoritmo es no lineal pero se

puede implementar muy eficientemente usando un filtrado lineal de las negativas log-verosimilitudes)

Capítulo 5 - Resultados y Discusión: Se presenta una comparación cuantitativa del algoritmo propuesto contra el estado de arte sobre un *benchmark*. También se muestra la capacidad del algoritmo para segmentar exclusivamente texturas.

Capítulo 1

Introducción

La habilidad para segmentar una o más imágenes presentadas a un sistema de visión (biológico, óptico, electrónico, etc.) es un paso fundamental para entenderla(s). La segmentación de una imagen es el proceso de definir las regiones y/o fronteras de uno o más objetos de interés en una imagen digital para poder separarlos entre ellos y del fondo. La meta final de muchos sistemas de visión computacional es "entender" una imagen, o un grupo de imágenes, al crear una representación descriptiva de la(s) imágen(es).
La segmentación continúa siendo un problema fundamental para la comunidad de visión computacional. La abundante investigación reciente refleja la opinión de que una segmentación precisa y eficiente es esencial para técnicas de alto nivel en una gran variedad de disciplinas y que pequeños avances pueden tener efectos de largo alcance. Mucho del trabajo en segmentación de imágenes, y visión computacional en general, continua enfocándose en técnicas automáticas y por lo tanto se asumen ciertas características en la imagen. Sin embargo, todas las técnicas de segmentación requieren de alguna forma el auxilio humano. Por lo tanto el objetivo de cualquier herramienta para identificar algún objeto debería maximizar la información de alto nivel proporcionada por un experto.
Dado que esta tesis se basa en la segmentación de imágenes resulta pertinente explicar brevemente lo que es la segmentación y su clasificación.

1.1. Segmentación

La Segmentación de una imagen particiona el dominio de una imagen en subconjuntos disconjuntos. De forma ideal las regiones resultantes deberían corresponder exactamente con varios de los objetos que se encuentran en la imagen. Desafortunadamente, actualmente, no existen técnicas de segméntación completamente automática que logre este objetivo para una clase general de imágenes. En la práctica, algoritmos de segmentación automática particionan una imagen de tal forma que las segmentaciones obtenidas generalmente exhiben una heterogeneidad intra-región y una homogeneidad intra-región [27]. La homogeneidad está típicamente basada en cierto criterio de uniformidad como la intensidad del pixel, el color, la textura, etc.

Más aún, algunos algoritmos de segmentación con frecuencia buscan definir, geométricamente y topológicamente, ciertas regiones - las fronteras deberían ser sencillas y las regiones deben evitar encontrarse agujeradas -.
La aplicación de estos criterios pueden partir automáticamente una imagen en múltiples segmentaciones homogéneas, sin embargo rara vez una región individual coincide exáctamente con un objeto de la imagen. En su lugar, un objeto está típicamente definido por múltiples regiones conectadas. Por lo tanto, muchas estrategias de segmentación de forma jerárquica fusionan regiones vecinas de forma sucesiva en otras más grandes con la meta de que el objeto de interés corresponda a una única región (fig 1.1).
Lamentablemente, aún las aproximaciones jerárquicas fallan con frecuencia al inten-

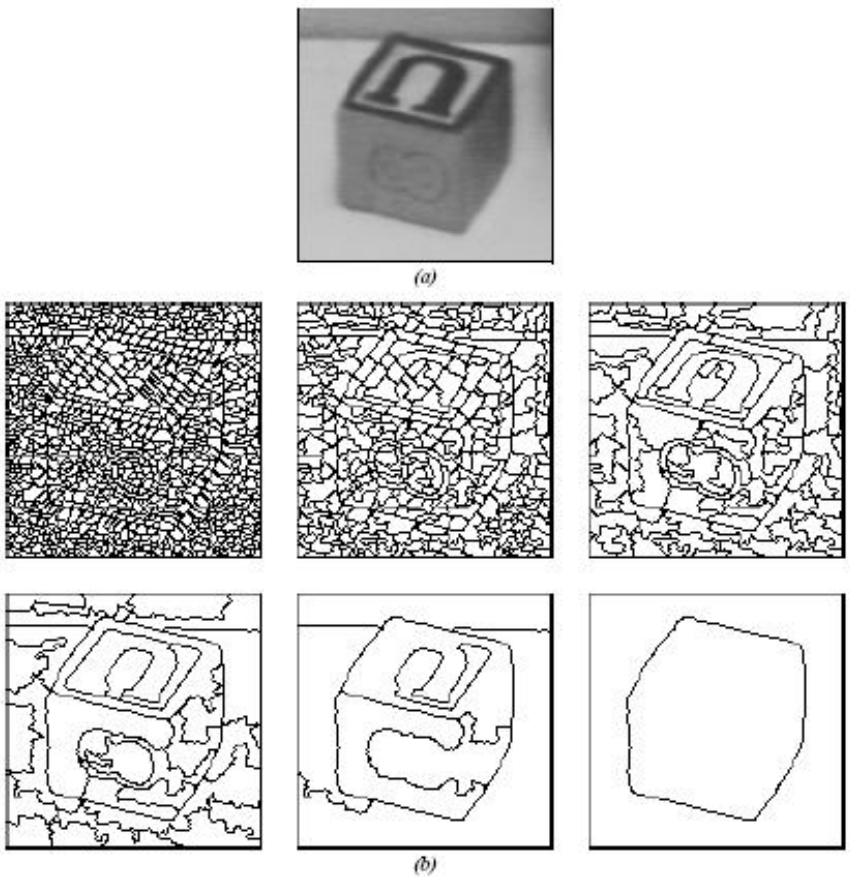

Figura 1.1: Segmentación jerárquica automática. (a)Imagen en nivel de grises de un bloque (128x128). (b)Niveles consecutivos de segmentación jerárquica (Crédito: [27])

tar aislar un objeto como una región. Estos métodos no logran segmentar todos los objetos y/o subobjetos dentro de una clase general de imágenes.
Aunque muchas técnicas especializadas [4] [18] [41] [47] segmentan correctamente objetos específicos dentro de un escenario limitado sin guía humana, estos típicamente

se basan en modelos de objetos *a priori* capturados bajo un rango restringido de iluminación. La segmentación completa (encontrar las regiones disjuntas que identifican únicamente todos los objetos de interés) no puede ser lograda sin un proceso de alto nivel [37], esto es sin especificar un dominio específico heurístico, modelos estadísticos/geométricos, guía humana, o cualquier otro tipo de supervisión. De estos, solamente la guía humana proveé una solución general para problemas de segmentación específicos.
Las estrategias de segmentación que se basan en los datos suministrados por el usuario inicializan un punto o región semilla, proveen de un contorno aproximado, ajustan parámetros de segmentación, etc. guían el proceso de segmentación a la meta deseada, son clasificados como herramientas de selección. En lugar de particionar automáticamente una imagen en múltiples regiones homogéneas, la herramientas de selección le permiten al usuario definir de forma interactiva objetos de interés y separarlos del resto del fondo no deseado. Más aún, los objetos seleccionados y/o el fondo con frecuencia no exhiben propiedades de homogeneidad identificables, más sólo características de alto nivel que las regiones seleccionadas pertenecen a los objetos identificados por el usuario.

1.2. Clasificación de técnicas de segmentación

La mayoría de los algoritmos de segmentación caen en alguna de estas tres categorías: (1) clasificación de pixel (basado en características), (2) manipulación de regiones (basado en regiones), y (3) manipulación de fronteras (basado en bordes). Como es típico, cada estrategia tiene sus ventajas y sus desventajas y se adapta mejor para resolver ciertos tipos de problemas.

1.2.1. Segmentación basada en características

Los métodos de clasificación de pixel son también conocidos como segmentación basada en características. Estas estrategias involucran clasificar pixeles basados en su posición en el espacio de características sin considerar explícitamente su conectividad a pixeles clasificados de forma similar. Algunas características son intensidad, color, magnitud de gradiente, textura, profundidad, movimiento, etc.
Aplicar un umbral en escala de grises es tal vez la técnica de segmentación más simple, basada en características o cualquier otra, dado que se basa únicamente en la intensidad del pixel. Un pixel se clasifica como parte de un objeto si su intensidad es mayor que o igual que un umbral de intensidad dado (fig 1.2). El umbral de intensidad puede fijarse de forma interactiva o automática [6]. Los umbrales automáticos pueden calcularse de forma global (para la imagen entera) [28] o de forma adaptativa (dentro de una ventana) [15], óptimamente [44] o ad-hoc [27]. Mientras que establecer un umbral es rápido y simple, está limitado a segmentar objetos que no se sobreponen en el espacio de características con el fondo u otros objetos de la imagen.
La clasificación basada en distancia asigna a cada pixel con la clase de objeto más cercana. Cada clase tiene un vector de características representativo - usualmente

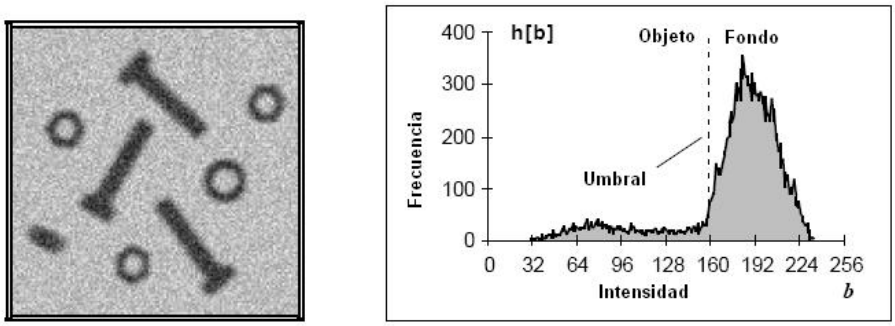

Figura 1.2: Los pixeles abajo del umbral serán etiquetados como parte del objeto y aquellos arriba de este se considerarán parte del fondo (Crédito: [30])

llamado centro del cluster - y cada pixel se asigna a la clase que minimiza la distancia al vector de características del pixel. Como en la aplicación del umbral, un vector de características puede incluir intensidad, color, gradiente de la imagen, textura, etc. La distancia mínima puede ser la Euclidiana, o puede ser en términos de varianza y covarianza como en la clasificación Bayesiana [59]. Los vectores de características representativos (i.e, los centros del cluster) pueden especificarse por un operador humano o pueden determinarse de forma automática via un algoritmo de clustering [59].

1.2.2. Segmentación basada en regiones

Un incoveniente de las técnicas basadas en características es que típicamente no consideran la relación espacial entre pixeles o la conectividad entre pixeles clasificados de forma similar. Por lo tanto, la clasificación basada en características puede ser disjunta, aún al extremo donde varios pixeles son clasificados de forma aislada. Es por lo anterior que los métodos basados en características con frecuencia requieren un paso de post-proceso para "limpiar" la segmentación [49] [25] [45]. Las estrategias basadas en regiones, por otro lado, intentan mantener la conectividad al agrupar pixeles con características similares. En esencia, los métodos basados en regiones pueden verse como una extensión de los basados en caaracterísticas al incluir la conectividad.
Etiquetar componentes conectados o *blob coloring* [6] es la técnica más simple basada en regiones. Dada una imagen binaria de multiples regiones desconectadas, *blob coloring* asigna a cada región una diferente etiqueta o color. Una técnica para *blob coloring* escanéa la imagen por filas (de izquierda a derecha y de arriba hacia abajo) y asigna a cada pixel del fondo una etiqueta basada en la(s) etiqueta(s) de los vecinos previamente procesados. Si un pixel no tiene ningún vecino procesado, este toma una nueva etiqueta. Si los vecinos de un pixel tiene más de una etiqueta equivalente, el pixel es etiquetado con una de las etiquetas (usualmente la etiqueta con el menor número) y el resto de la etiquetas se fijan equivalentes. Un segundo paso re-etiqueta pixeles etiquetados como equivalentes a la misma etiqueta.
El crecimiento de una región comienza con una o más semillas y estas crecen al incluir pixeles vecinos a una región existente basándose en algún criterio de homogeneidad.

La semillas pueden ser pixeles individuales (u otras formas de regiones predefinidas); aún cuando las semillas pueden inicializarse de forma automática [55], el crecimiento de regiones se hace frecuentemente sembrando manualmente. El criterio para incluir un pixel vecino a una región existente puede incluir características basadas en bordes como la magnitud del gradiente al igual que características basadas en regiones como la diferencia de la varianza ajustada de intensidad/color de la actual región y el límite puede ser muy difícil o "fuzzy" [40]. Cuando más de una semilla es especificada, regiones mutuamente exclusivas compiten por espacio en la imagen y tienden a reducir fugas a través de fronteras débiles de los objetos [52]. Finalmente, el crecimiento de regiones puede ser fácilmente extendido para datos tridimensionales como los que se presentan en el análisis de imágenes médicas [57].
Mientras el crecimiento de regiones comienza con pocas regiones (o tal vez solamente una), la fusión de regiones [37] comienza al asumir cada pixel - o pequeñas regiones - como una región individual y se fusiona de forma jerárquica con regiones vecinas que tienen propiedades similares. Si una región no tiene ningún vecino que satisface el criterio de similaridad, la región es marcada como final y la fusión termina cuando todas las regiones son finales.
Una ventaja que tiene la fusión de regiones sobre el crecimiento de regiones es que como regiones con orden jerárquico se vuelven más y más grandes de forma progresiva, hasta ser lo suficientemente grandes para exhibir propiedades de textura que pueden utilizarse como criterio para fusionar regiones de menor jerarquía. Tales propiedades serían dificiles de calcular para los pixeles individuales usados en el crecimiento de regiones. Además, mientras el crecimiento de regiones es usualmente inicializado de forma interactiva, la fusión de regiones es típicamente un proceso totalmente automático.
Contrario al estilo de fusión de regiones se encuentra la división de regiones cuya aproximación para segmentar empieza con la imagen entera como una sola region y de forma recursiva la divide hasta que cada región encontrada satisface un criterio de homogeneidad. Las regiones pueden dividirse de forma uniforme usando un *quadtree* [46], de forma no uniforme vía triangularización [26], o de forma arbitraria suando *graph cut* (fig 1.3) [61] o una técnica basada en árboles [62].

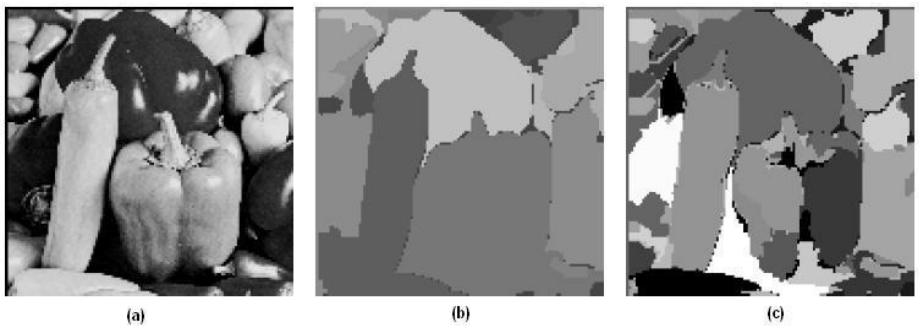

Figura 1.3: Segmentación basada en regiones por *nested cuts* (a)Imagen original (b)Máxima jerarquía (c)Mínima jerarquía (Crédito: [61])

1.2.3. Segmentación basada en bordes

Mientras los métodos basados en regiones buscan identificar los grupos de pixeles conectados que definen varios objetos de la imagen, las estrategias basadas en bordes o fronteras buscan definir los contornos que encierran estos objetos o subobjetos. Las representaciones de los basados en bordes y de los basados en regiones son duales de cada quien dado que es fácil detectar bordes de regiones existentes y de igual forma contruir regiones a partir de contornos cerrados. Sin embargo, aún con la dualidad de las representaciones, los resultados producidos por las dos aproximaciones difieren usualmente debido a que los métodos basados en bordes utilizan típicamente distintos criterios que las técnicas basadas en regiones.
La más sencilla de las técnicas basadas en bordes es el seguimiento del contorno [6]. Esta estrategia produce resultados idénticos que los obtenidos por los basados en características con umbral con la excepción que el seguimiento de contorno genera una frontera que encierra una región en lugar de identificar cada pixel de la región. Una técnica popular de seguimiento de contorno escanea la imagen para un pixel arriba del umbral de intensidad dado. Una vez encontrado, emplea una búsqueda denominada "sobre el hombro", la cual consiste en rastrear alrededor de una región de la imagen mayor que (o igual que) el valor del umbral [6]. Se marca cada pixel en la línea divisora encontrada y después continua el escaneo para otra región.
El seguimiento local de bordes es similar al seguimiento del contorno sólo que en lugar de buscar un umbral de intensidad, los algoritmos de seguimiento de bordes buscan por un punto borde que cumple lo mejor posible con las características del actual borde [8]. Dado un punto borde y una dirección de búsqueda, se encadena a un punto de una vecindad que cumple lo mejor posible el criterio del borde. El seguimiento local de bordes es por lo tanto susceptible a posibles fallas cuando se presentan cambios drásticos en las características de la frontera. Debido a su naturaleza local es también susceptible al ruido y es propenso a confundirse fácilmente con intersecciones de bordes.
La relajación de bordes es otro algoritmo local basado en fronteras. Es un proceso iterativo que empieza con un conjunto de segmentos de frontera candidatos. Cada punto segmento de frontera tiene una dirección y una probabilidad o verosimilitud de ser un punto frontera. Un algoritmo de relajación ajusta, de forma iterativa, la verosimilitud de un punto basándose en la verosimilitud de sus vecinos y dirección. Dado que estrictamente es un método local es susceptible a caer en mínimos locales. Más aún, debido a su naturaleza iterativa, la relación de bordes no es adecuado cuando existe interacción.
La programación dinámica (DP) [9] es una técnica popular usada para encontrar las fronteras globales óptimas basada en un criterio de pesos locales [7] [13] [14]. La programación dinámica puede formularse como un problema de grafos [39] [38]. El objetivo es encontrar la trayectoria óptima global en un grafo direccionado. Tradicionalmente, DP y búsqueda en grafos es computacionalmente muy complejo, por lo tanto típicamente, se restringe la búsqueda a un submuestreo en 1-D de una imagen 2-D. Técnicas óptimas automáticas de detección de bordes han sido desarrolladas

para varios tipos de imágenes específicas [4] [18] [41].
Entre las técnicas globales basadas en bordes, se encuentra contornos activos (también conocidos como snakes) que han recibido mucha atención [3] [17] [23]. Típicamente, los contornos activos son inicializados de forma manual con una aproximación burda de la frontera de interés. El algoritmo actúa de forma iterativa sobre la frontera para determinar la frontera que minimice un funcional de energía. El funcional de energía es una combinación de energía externa suministrada por la imagen (como la magnitud del gradiente y dirección) y de energía interna (tal como la curvatura de la frontera y la distancia entre puntos de control). Finalmente, los contornos activos pueden ser representados de forma explícita con contornos paramétricos o de forma implícita como *level set* (fig 1.4) [48].

Figura 1.4: Evolución de la segmentación basada en bordes por *level sets* de un arbol arterial (Crédito: [48])

Capítulo 2

Segmentación asistida

La segmentación automática de cualquier tipo de imágenes sigue siendo un problema sin resolverse, esto se debe a la gran variedad de fuentes, contenido y complejidad de las mismas. Esto ha conducido, como puede apreciarse en el capítulo anterior, a una gran variedad de aproximaciones semiautomáticas, esquemas de inicialización, etc. En muchos de los casos, la segmentación manual (i.e., trazar las fronteras de los objetos) sigue siendo usada comunmente cuando el componente de una imagen debe ser segmentada de un fondo complejo. Es por esta razón que herramientas de segmentación inteligentes que explotan la información de alto nivel de expertos y requieren poca interacción con el usuario se han vuelto muy atractivas.
La asistencia en el proceso de segmentación resulta parte fundamental de este trabajo por lo que se presentarán las propiedades deseadas de las técnicas de segmentación asistida al igual que la opción elegida, que son el uso de trimapas y su justificación.

2.1. Propiedades deseadas para técnicas de segmentación asistida

La mayoría de las herramientas para seleccionar objetos caen de alguna forma entre visión asistida por el usuario e interacción basada en visión [42]. La visión asistida por el usuario describe técnicas donde el usuario interactúa con la imagen para empezar y/o guiar un algoritmo de visión para que produzca el resultado deseado. Por ejemplo, *magic wand* [29] calcula regiones conectadas de pixeles similares basándose en el click de una área de interés. La interacción basada en visión se refiere a aquellos métodos donde la computadora ha hecho algo o todo de la parte de visión y el usuario interactúa dentro del espacio de características resultado. Un ejemplo es el sistema ICE (Interactive Contour Editing) [21] que calcula la representación de bordes de una imagen y después le permite al usuario seleccionar de forma interactiva grupos de bordes para remover características de la imagen.
Una herramienta es clasificada basándose en cuando el usuario interactúa con el algoritmo. La visión asistida por el usuario manipula el espacio de entrada (dominio) de la función de visión mientras que la interacción basada en visión provee acceso

al resultado (rango). Mientras que los métodos de visión asistida por el usuario casi siempre interactúan en términos de la imagen (dominio), los datos del usuario para las técnicas basadas en visión pueden ocurrir al manipular directamente el rango o remapeando los datos dados a través del dominio para modificar el resultado dentro del rango de la función de visión. Algunas herramientas permiten la intervención en varios pasos en el proceso, incluyendo la habilidad para ajustar los resultados parciales o intermedios.
Sin importar la clasificación de la herramienta, existen algunas propiedades algorítmicas que son deseables para las técnicas de selección de objetos. Estas propiedades incluyen:

- *Simple*: Mientras que los cálculos pueden ser complejos, la interfaz presentada al usuario debe ser sencilla e intuitiva.
- *Predecible*: El usuario puede generalmente anticiparse a la respuesta de la herramienta dada cierta imagen, aún cuando no esté familiarizado con el algoritmo.
- *Robusto*: Que sea capaz de aceptar imágenes ruidosas y aún así producir el resultado deseado.
- *Inteligente*: "Entiende" o "aprende" que es lo que el usuario quiere, mientras menor esfuerzo y tiempo se requiera del usuario para manipular la herramienta y obtener el resultado deseado, más inteligente será la herramienta.
- *Interactiva*: La habilidad para interactuar con la herramienta en proceso.
- *Rápida*: El usuario no debería esperar entre datos interactivos.
- *General*: Se tiene una amplia aplicabilidad y puede ser usado efectivamente en una gran variedad de imágenes.
- *Manipulable*: La habilidad para forzar un comportamiento específico deseado (dentro de límites razonables dado el marco de la herramienta) o modificar fácilmente el resultado para alcanzar los resultados deseados.

No es necesario para una herramienta el poseer todas las propiedades para ser efectivo. Aunque algunas propiedades necesitan de otras. Por ejemplo, mientras más inteligente sea una herramienta, se requerirá mayor manipulación cuando "inteligentemente" actúa erroneamente.
Una propiedad que es muy importante en casi cualquier herramienta es que sea predecible. La predecibilidad está influenciada no solamente por la función de visión y sus propiedades asociadas, pero también por la herramienta de clasificación. La interacción basada en visión tiende a ser más predecible dado que el proceso de visión ha sido completado antes de que ocurra la interacción. Por lo tanto, el mapeo entre la información del usuario y los resultados es, potencialmente, más inmediata y sencilla. Sin embargo, en las herramientas de visión asistida por el usuario, la información del usuario tiene que pasar a través de la función de visión, lo cual puede ser my complejo y por lo tanto confundir el mapeo. Sin embargo, eso no implica que las herramientas asistidas por el usuario sean impredecibles.

2.2. Trimapa

Entre las herramientas de visión asistida por el usuario se encuentran los trimapas que como su nombre lo indican son un mapa de etiquetas sobre clases de objetos de interés, las etiquetas especifican 3 tipos de clases diferentes, estas son *objeto*, *fondo* y *zona desconocida*. Un trimapa indica con certeza donde se localiza el objeto que nos interesa y aquello que no nos interesa, dejando una zona de incertidumbre. Como

Figura 2.1: Ejemplo de un trimapa donde se señala el objeto de interés que en este caso es el sombrero y el resto de la imagen

puede observarse en (fig 2.1) el trimapa consiste en que el experto seleccione el objeto de interés de un color y el resto de otro, después algún algoritmo se encargará de realizar la segmentación tomando en cuenta la información proporcionada por el trimapa. La extensión a multimapas es muy sencilla sólo se selecciona de otro color aquellos objetos de interés.
Los trimapas se han usado para hacer un campo de transparencias en imágenes con el objetivo de insertar el objeto de interés en otra imagen correspondiente a una escena diferente [12] [2]. Esta idea surgió de la clásica pantalla verde que se utiliza en el cine para extraer de manera sencilla el fondo y poder así crear escenarios que de no ser por ese medio no hubieran podido llevarse a la pantalla grande por ser prácticamente

imposibles o muy costosas.
Los trimapas serán la herramienta que se utilizará en este trabajo para nuestro algoritmo de segmentación asistida debido a que cumplen con varias de las propiedades deseadas antes mencionadas. Resultan *simples* porque el usuario sólo debe rayar los objetos que quiera segmentar de forma sencilla como una aplicación tipo *Paint*, *predecibles* porque dependiendo de la complejidad de la imagen el usuario con el tiempo podrá identificar cuanta información será conveniente proporcionar a través del trimapa independientemente del algoritmo que la trabaje, *generales* porque se pueden aplicar de forma efectiva para cualquier tipo de imagen y *manipulables* porque pueden modificarse de forma sencilla, haciendo más específico al trimapa, hasta alcanzar el resultado deseado.

Capítulo 3

Campos aleatorios Markovianos

Con el artículo [54] en 1984 se introdujeron los campos Markovianos aleatorios, utilizados anteriormente por físicos y estadísticos, de forma comprensiva y estimulante como poderosas herramientas en el marco de procesamiento de imágenes. Desde entonces se ha explotado su teoría y versatilidad en diversas áreas como modelación de imágenes (para propósitos de síntesis, reconocimiento o compresión) o con la solución de problemas inversos de alta dimensión en visión temprana (como clasificación y segmentación).
La suposición implícita detrás de las aproximaciones probabilistas para el análisis de imágenes es que, para un problema dado, existe una distribución de probabilidad que puede capturar la variabilidad y las interacciones entre diferentes conjuntos de características relevantes de la imagen.
En este capítulo se dará una breve introducción de los campos Markovianos aleatorios y se explicarán los *campos Markovianos de probabilidad cuadráticos* que se utilizan para la segmentación de imágenes y es la principal herramienta de este trabajo.

3.1. Definición

Sea $F = \{F_r | r \in \mathcal{S}\}$ un conjunto de variables aleatorias definidas sobre el conjunto de sitios $\mathcal{S}$, donde cada variable aleatoria F_r toma el valor $f_r \in L$ donde L es un conjunto de etiquetas. Llamamos a F, un *campo aleatorio*. Denotamos por $P(F_r = f_r)$, la probabilidad de que la variable aleatoria F_r tome el valor f_r, y la abreviamos $P(f_r)$. Denotamos también la probabilidad conjunta de que el campo aleatorio F tome el valor de la configuración f por $P(F = f)$ y la abreviamos $P(f)$.
Sea $\mathbb{F}$ el conjunto de todas las posibles configuraciones de los sitios S con etiquetas L. Definimos un *campo aleatorio markoviano* respecto al sistema de vecindades $\mathcal{N}$, como un campo aleatorio que cumple las siguientes propiedades:

- $P(f) > 0 \ \forall f \in \mathbb{F}$
- $P(f_r | f_{S-\{r\}}) = P(f_r | f_{\mathcal{N}_r})$ donde $f_{S-\{r\}} = \{f_s | s \in \mathcal{S}\}$ y $f_{\mathcal{N}_r} = \{f_s | s \in \mathcal{N}_r\}$

La última propiedad nos indica que un campo aletorio será Markoviano si cada variable aleatoria del campo es independiente de todas las variables aleatorias que se

encuentren fuera de su vecindad $\mathcal{N}_r$. Una vez definida la relación que debe existir entre los sitios de un campo aleatorio para que sea Markoviano, lo que sigue es determinar una manera de construir funciones de distribución adecuadas. A continuación se presenta la única forma de lograrlo.

3.2. Campos aleatorios de Gibbs

Un conjunto de variables aleatorias F es un *campo aleatorio de Gibbs* si y sólo si para cada configuración $f \in \mathbb{F}$:

$$P(f) = \frac{1}{Z} \exp^{-U(f)} \tag{3.1}$$

donde $U(f)$ se conoce como la *función de potencial* y se define como

$$U(f) = \sum_{c \in \mathcal{C}} V_c(f) \tag{3.2}$$

que es la suma de funciones de potencial sobre todos los "cliques" (o pandillas) asociados al sistema de vecindades dado.
A esta distribución se le conoce como *distribución de Gibbs*. Dado que P debe ser una función de densidad, es claro que

$$Z = \sum_{f \in \mathbb{F}} \exp^{-U(f)}$$

actúa simplemente como una constante de normalización.

El **Teorema de Hammersley-Clifford** [35] establece que

> Un campo aleatorio F sobre un conjunto de sitios $\mathcal{S}$, es Markoviano respecto a un sistema de vecindades $\mathcal{N}$ si y sólo si F es un campo aleatorio de Gibbs sobre el conjunto de sitios S, respecto al sistema de vecindades $\mathcal{N}$.

La importancia del Teorema anterior es que permite construir funciones de densidad para campos aleatorios (de manera global) fijando las funciones de potencial para el conjunto de cliques (de manera local).

3.3. Estimación Bayesiana

Sea $f \in \mathbb{F}$ una observación de un fenómeno o el resultado de un experimento dado (a $\mathbb{F}$ se le conoce como el *espacio muestral* de la variable aleatoria f) y supongamos que dicha observación responde a una distribución dada $\eta(\theta)$, es decir, $f \sim \eta(\theta)$ donde θ es un conjunto de parámetros desconocidos cuyos valores pertenecen a un espacio de parámetros Ω. El problema de la estimación consiste en encontrar el *estimador* $\hat{\theta}$ que se ajuste lo mejor posible al valor de θ bajo algún criterio de *optimalidad*. Para

esto, se fija una *función de costo* $C(\theta^*, \theta)$ que mide el nivel de error cometido al elegir como estimador a θ^* dado que el valor verdadero es θ. Note que cualquier medida de distancia podría utilizarse como función de costo, por ejemplo $C(\theta^*, \theta) = \|\theta^* - \theta\|^2$. Dado que la definición de la función de costo depende del valor verdadero de los parámetros que se desconocen no es posible evaluar dicha función. Sin embargo se puede calcular la esperanza del costo dado f:

$$E\left[C(\theta^*, \theta)|f\right] = \int_{\theta\in\Omega} C(\theta^*, \theta)P(\theta|f)d\theta$$

lo que reduce el problema a encontrar un estimador θ^* que minimice esa expresión.
La probabilidad $P(\theta|f)$ se conoce como *probabilidad a posteriori* y la función de distribución subyacente se conoce como *distribución a posteriori*. Para calcular $P(\theta|f)$ se utiliza la regla de Bayes

$$P(\theta|f) = \frac{P(f|\theta)P(\theta)}{P(f)}$$

A la probabilidad $P(f|\theta)$ se le llama *función de verosimilitud*, la cual es conocida una vez que se conoce la distribución η y que se ha fija algun valor para θ (note que la observación f se encuentra fija). La probabilidad $P(\theta)$ se conoce como *probabilidad a priori* y nuevamente, la distribución subyacente es la *distribución a priori*. Finalmente $P(f)$ es una expresión que no depende del valor de θ ni de la elección de θ^*.
En el marco de la estadística Bayesiana una vez que se conoce la función de verosimilitud y la distribución *a priori*, la mejor estimación que se puede obtener es el *estimador Bayesiano óptimo*.
Las elecciones más populares para la función de costo son:

- $C(\theta^*, \theta) = \|\theta^* - \theta\|^2$
- $C(\theta^*, \theta) = 1 - \delta(\theta^* - \theta)$

y los estimadores Bayesianos óptimos correspondientes (minimizadores de la esperanza del costo) son:

- $\theta^* = \int_{\theta\in\Omega} \theta P(\theta|f)d\theta = \bar{\theta}$
- $\theta^* = \arg\max_{\theta\in\Omega} P(\theta|f)$

Este último se conoce como *estimador maximizador de la distribución a posteriori* (MAP) [35].

3.4. Campos Markovianos de probabilidad cuadráticos

Lo que se presenta a continuación proviene del trabajo de Rivera *et al.* [50] quienes propusieron los modelos de Campos Markovianos de Probabilidad Cuadráticos (QMPF por sus siglas en inglés) para la Segmentación Binaria de Imágenes; entrado

en la categoría de segmentación basada en características. Para poder segmentar K regiones de interés de una imagen g, se realiza la siguiente representación

$$g(x) = \sum_k \alpha_k(x) g(x) \tag{3.3}$$

para $k = 1, 2, \cdots, K$, donde

$$\alpha_k(x) g(x) = \alpha_k(x)(I_k(x) + \eta_k(x)) \tag{3.4}$$

donde $x \in \mathcal{R} \subseteq \mathcal{L}$ denota la posición de un pixel en la región e interés, $\mathcal{R}$, en el mallado regular $\mathcal{L}$. Las I_k son las imágenes de composición, η_k es una imagen de ruido con una distribución conocida y los α_k se denominan *matting factors*, lo cuales deben satisfacer:

$$\begin{aligned}
\sum_{k=1}^{K} \alpha_k(x) &= 1, \qquad x \in \mathcal{R} && (3.5)\\
\alpha_k(x) &\geq 0, \qquad k = 1, \cdots, K && (3.6)\\
\alpha_i(x)\alpha_j(x) &\approx 0, \qquad \text{si } i \neq j && (3.7)\\
\alpha(x) &\approx \alpha(y), \qquad y \in \mathcal{N}_x && (3.8)
\end{aligned}$$

donde $\mathcal{N}_x$ denota al conjunto de vecinos de x como $\mathcal{N}_x = \{y \in \mathcal{R} : |x - y| = 1\}$. Note que, por (3.5) y (3.6), α puede ser interpretado como un Campo de Probabilidad donde $\alpha_k(x)$ se entiende como la probabilidad de producir la observación $g(x)$ usando la imagen $I_k(x)$. Por otro lado la ecuación (3.7) introduce la restricción de baja entropía en el vector de probabilidad $\alpha(x)$. Juntando las ecuaciones (3.5) y (3.6) y la restricción (3.7) nos indican que solamente una entrada del vector $\alpha(x)$ tiene un valor cercano a uno mientras los otros tienen valores cercanos a cero. La restricción (3.8) promueve que el campo de probabilidad α sea suave en el espacio.

La segmentación de la imagen compuesta, g, puede observarse como la solución de un problema inverso mal planteado en (3.3) y (3.4) sujeto a las restricciones *fuertes* (3.5) y (3.6) y a las restricciones *suaves* (3.7) y (3.8). Dicho problema inverso puede entenderse como el cálculo de los *matting factors* α_k y las imágenes originales I_k, o al menos las imagenes parciales $\alpha_k I_k$ a partir de la imagen observada g.

En el marco de Regularización Bayesiana (BR por sus siglas en inglés), uno calcula la solución (α^*, I^*) como un estimador de la distribución *a posteriori* $P(\alpha, I|g)$. Como en la sección anterior se explicó, esta distribución posterior puede calcularse usando la regla de Bayes como:

$$P(\alpha, I|g) = \frac{1}{Z} P(g|\alpha, I) P(\alpha, I), \tag{3.9}$$

donde $P(g|\alpha, I)$ es la probabilidad condicional de los datos al asumir dados (α, I), $P(\alpha, I)$ es la distribución *a priori* y $Z = P(g)$ es una constante de normalización (independiente de (α, I)). En este contexto, la probabilidad condicional $P(g|\alpha, I)$ se deriva de la distribución de la distribución del ruido y de las ecuaciones (3.3) y (3.4);

la distribución *a priori* $P(\alpha, I)$ codifica nuestro conocimiento *a priori* acerca de la Markovianidad de los parámetros y su baja entropía.
En general, la inferencia de las imágenes I_k a partir de g es un problema inverso complejo: aún si α es dada, sólo se podrían recuperar las fracciones $\alpha_k I_k$ de las imágenes completas. Por lo tanto es importante realizar ciertas suposiciones, como considerar que tales imágenes pueden ser representadas por una función paramétrica $I_k(x) = \Phi(x, \beta)$, con parámetros β. Para simplificar la notación el desarrollo a continuación se hará en términos de I en lugar de los parámetros β. Otra forma popular de introducir conocimiento *a priori* de alto nivel para segmentar imágenes con escenarios complejos es la interacción con el usuario. Bajo ese esquema el usuario etiqueta a mano unos subconjuntos de pixeles, para posteriormente etiquetar aquellos que no lo son con un algoritmo de segmentación que toma en cuenta la distribución de los pixeles etiquetados y la suavidad de las regiones a segmentar.
Para obtener $P(g|\alpha, I)$ se asume que η_k es ruido Gaussiano i.i.d. con media cero y varianza σ_k^2, i.e.:

$$P(\eta_k(x)) = G_{\sigma_k}(\eta_k(x)) \tag{3.10}$$

donde

$$G_\sigma(z) := \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{z^2}{2\sigma^2}\right]. \tag{3.11}$$

De (3.4) se tiene $\alpha_k(x)\eta_k(x) = \alpha_k(x)(g(x) - I_k(x))$. Como α_k es casi binaria [por (3.8)], entonces para $\alpha_k(x) \approx 1$ uno puede esperar una distribución similar tanto de $\alpha_k(x)\eta_k(x)$ y $\eta_k(x)$. Por lo tanto, definiendo $r_k(x) = g(x) - I_k(x)$ se tiene:

$$\begin{aligned} P(\alpha_k g|\alpha_k I_k, \sigma_k^2) &= \prod_x G_{\sigma_k}(\alpha_k(x) r_k(x)) \\ &= \prod_x G_{\sigma_k}(r_k(x))^{\alpha_k^2(x)}. \end{aligned} \tag{3.12}$$

Ahora se removerá la suposición de ruido Gaussiano. Primero se debe considerar que cualquier distribución de densidad suave v_k puede ser expresada como un mezcla de Gaussianas:

$$v_k(x, \theta_k) = \sum_{i=1}^{M} \pi_{ki} G_{\sigma_k}(r_k(x) - m_{ki}) \tag{3.13}$$

con $\theta = (\sigma_k, \pi_k, m_k)$; donde $\pi_{ki} \geq 0$ son los coeficientes de la mezcla (con $\sum_i \pi_{ki} = 1$), M es el número de Gaussianas de la mezcla, $m_k = (m_{k1}, m_{k2}, \cdots, m_{kM})$ son las medias de las Gaussianas y σ_k son sus varianzas. Se asume que los parámetros de la mezcla son conocidos, i.e. se asume la distribución de ruido conocida. Entonces en

límite de baja entropía se tiene:

$$\begin{aligned} P(\alpha_k g|\alpha_k I_k, \theta_k) &= \prod_x \left[\sum_{i=1}^{M} \pi_{ki} G_\sigma(r_k(x) - m_{ki})^{\alpha_k^2(x)}\right] \\ &\approx \prod_x \left[\sum_{i=1}^{M} \pi_{ki} G_\sigma(r_k(x) - m_{ki})\right]^{\alpha_k^2(x)} \\ &= \prod_x v_k(x, \theta_k)^{\alpha_k^2(x)} \end{aligned} \tag{3.14}$$

Si se asume independencia entre $\alpha_i I_i$ y $\alpha_j I_j$ (para $i \neq j$), entonces la verosimilitud de g, la imagen compuesta, esta dada por

$$P(g|\alpha, I, \theta) = \prod_k P(\alpha_k g|\alpha_k I_k, \theta_k) \tag{3.15}$$

En particular, tal independencia ocurre si se satisface (3.7). Para promover la baja entropía, proponemos usar el potencial $\mu(1 - \sum_k \alpha_k^2(x))$ (con $\mu > 0$ se promueve baja entropía), conocido como coeficiente de Gini. Además una distribución Gibbsiana basada en modelos MRF controla la granularidad de las regiones (i.e. para promover regiones suaves). Finalmente se obtiene la distribución *a priori* como:

$$P(\alpha) = \frac{1}{Z'} \exp\left[\sum_{x \in \mathcal{R}} \left(\mu\left(1 - \|\alpha(x)\|^2\right) - \lambda \sum_{y \in \mathcal{N}_x} \|\alpha(x) - \alpha(y)\|^2\right)\right] \tag{3.16}$$

donde Z' es una constante de normalización. Al asumir independencia entre I y α se puede usar $P(\alpha, I) \propto P(\alpha)$ como probabilidad *a priori* y una distribución uniforme en I. Entonces la distribución *a posteriori* toma la forma $P(\alpha, \theta|g) \propto exp\left[-U(\alpha, \theta)\right]$ y el estimador MAP se calcula minimizando la función de costo:

$$U(\alpha, \theta) = \sum_{x \in \mathcal{R}} \left\{\sum_{k=1}^{K} \alpha_k^2(x) \left[-\log v_k(x, \theta) - \mu\right] + \frac{\lambda}{2} \sum_{y \in \mathcal{N}_x} |\alpha(x) - \alpha(y)|^2\right\} \tag{3.17}$$

sujeto a las restricciones (3.5) y (3.6). Este problema se puede resolver con cualquier estrategia de optimización con restricciones para problemas cuadráticos (programación cuadrática).

Es posible aún, introducir la información de que los bordes de las regiones coninciden, muchas veces, con los bordes de color o intensidad en la imagen, para ello se cambia el término de regularización en (3.17) por

$$\lambda \sum_{y \in \mathcal{N}_x} \left[\alpha(x) - \alpha(y)\right]^2 l_{xy} \tag{3.18}$$

donde $\mathcal{N}_x = \{y \in \mathcal{R} \cup \mathcal{B} \cup \mathcal{F} : |x - y| = 1\}$ para restringir que las regiones respeten los bordes de color. La información de los bordes de color se calcula con

$$l_{xy} = \frac{\gamma}{\gamma + \|g(x) - g(y)\|^2} \tag{3.19}$$

donde γ es el parámetro que controla la sensibilidad para detectar los bordes. En particular, en el caso en que la imagen g sea de colores, esta se transforma al espacio de color CIE-Lab con la librería de Ruzon [53]. l_{xy} puede interpretarse como una medida de afinidad que toma valores cercanos a uno si los pixeles vecinos x y y tienen colores similares y cerca de cero en el caso contrario.

Capítulo 4

Trabajo propuesto

La motivación de esta tesis consiste en incluir la textura como información extra para poder segmentar de forma interactiva las imágenes de manera más precisa. Cabe señalar que no se buscaba crear un algoritmo para segmentar exclusivamente texturas sino utilizar esta información para mejorar los resultados antes obtenidos. Resulta pertinente revisar los detalles del algoritmo anterior, para posteriormente explicar cuales fueron los cambios realizados. También se dará, en este capítulo, una breve introducción sobre texturas.

4.1. Segmentación de imágenes con QMPF y trimapas

El problema de segmentar una imagen g mediante QMPF se reduce a minimizar la función de costo (3.17) sujeta a las restricciones (3.5) y (3.6). Dado que se tiene una función cuadrática con restricciones lineales existen muchos métodos que pueden aplicarse (ver [43]) que garantizan la convergencia a algún mínimo local.
La elección para resolver el problema fue un esquema de actualización tipo Gauss-Seidel [49] que le permite al usuario obtener en cada iteración del algoritmo una mejor aproximación a la solución y además puede acelerarse la convergencia con una buena aproximación inicial (α^0).
Para obtener este esquema primero se construyó el Lagrangiano asociado a (3.17) tomando únicamente la restricción de igualdad (3.5), dado que este tipo de formulación nos garantiza la convergencia al mínimo global y requiere de menos cálculos, quedando

$$\max_{\pi} \min_{\alpha} \mathcal{L}(\alpha, \theta) = U(\alpha, \theta) - \sum_{x \in \mathcal{R}} \pi(x) \left(1 - \sum_{k} \alpha_k(x) \right) \tag{4.1}$$

$$\text{sujeto a } \alpha_k \geq 0$$

donde π son los multiplicadores de Lagrange correspondientes a las restricciones de igualdad. Ahora si se obtiene el gradiente de (4.1) con respecto a α y se iguala a cero se tiene la primer restricción KKT. Después para obtener un método iterativo tipo Gauss-Seidel se despeja $\alpha_k(x)$ y finalmente para eliminar los multiplicadores de

Lagrange y cumplir con la restricción de igualdad se sustituye en (3.5). La ecuación de actualización Gauss-Seidel queda:

$$\alpha_k(x) = \frac{n_k(x)}{m_k(x)} + \frac{1 - \sum_{l=1}^{K} \frac{n_l(x)}{m_l(x)}}{\sum_{l=1}^{K} \frac{m_k(x)}{m_l(x)}} \tag{4.2}$$

donde

$$n_k(x) = \lambda \sum_{y \in \mathcal{N}_x} (\alpha_k(y) * l_{xy}) \tag{4.3}$$

y

$$m_k(x) = (-\log v_k(x, \theta) - \mu) + \lambda \sum_{y \in \mathcal{N}_x} l_{xy} \tag{4.4}$$

Note, que el $\alpha_k(x)$ calculada en (4.2) no garantiza que se cumpla con la restricción de desigualdad (3.6). Si esta restricción es violada, es necesario regresar a $\alpha(x)$ a la región factible. Una forma sencilla de lograrlo es proyectar aquel $\alpha_k(x)$ negativo y renormalizar el vector $\alpha(x)$ para seguir satisfaciendo (3.5). Una vez alcanzada la convergencia de nuestro algoritmo, para obtener la segmentación dura basta con seleccionar al pixel x como parte de la región k si $\alpha_k(x)$ es el máximo de $\alpha(x)$.
Sólo resta comentar sobre el valor de $v_k(x)$ y la inicialización de $\alpha_k(x)$. La $v_k(x)$ es la verosimilitud que nos da la probabilidad de que el pixel x pertenezca a la región k. Para obtenerla se debe tener un modelo para cada k región que sea una distribución de probabilidad y nos permita evaluar cada pixel. Sin embargo para poder generar el modelo de una región en específico se hace uso del trimapa que mediante rayones realizados por expertos se definen tres regiones disjuntas que son:

- $\mathcal{B} \equiv \{x : x$ es un punto del fondo de la imagen $\}$,
- $\mathcal{O} \equiv \{x : x$ es un punto del objeto a segmentar $\}$,
- $\mathcal{U} \equiv \{x : x$ es un punto que no se sabe si $x \in \mathcal{B} \vee x \in \mathcal{O}\}$.

Es a partir de la información proporcionada por el trimapa que se puede calcular una distribución empírica basándose en los histogramas normalizados de cada una de las regiones. Entonces con estos modelos se pueden evaluar las $v_k(x)$ requeridas, por ejemplo, evaluando cualquier $x \in \mathcal{U}$ en el histograma obtenido de $\mathcal{O}$ se obtiene $v_{\mathcal{O}}(x)$. Finalmente una buena aproximación de $\alpha_k^0(x)$ es igualar con $v_k(x)$, dado que en cierta forma el cálculo de las verosimilitudes resulta una segmentación burda que se espera "limpiar" durante las iteraciones del esquema tipo Gauss-Seidel. En [50] se hace notar que la restricción de la sumatoria igual a uno puede ser eficientemente introducida, para el caso binario, en la función de costo usando una estrategia de eliminación de variables.
Englobando, el algoritmo queda de la siguiente manera:

1. Leer o crear el trimapa de la imagen a segmentar.
2. Calcular los histogramas de las k regiones a segmentar y normalizarlos.

3. Evaluar las verosimilitudes de cada pixel $x \in \mathcal{U}$.
4. Iniciar las α's con las verosimiltudes.
5. Elegir los valores de los parámetros μ, λ y γ.
6. Iterar (4.2) cuidando permanecer en la región factible hasta lograr la convergencia o alcanzar el número máximo de iteraciones.
7. Seleccionar al pixel x como parte de la región k si $\alpha_k(x)$ es el máximo de $\alpha(x)$.

En la figura (fig 4.1) se presenta un ejemplo de la segmentación realizada por el algoritmo antes presentado.

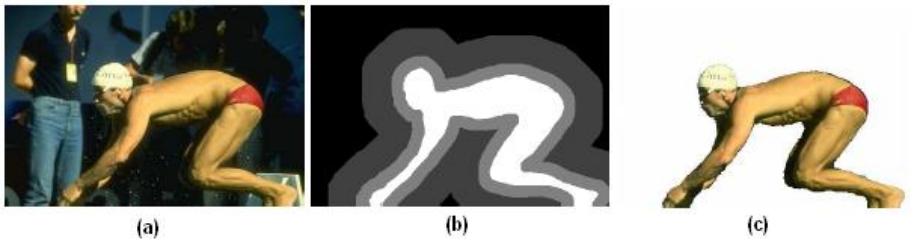

Figura 4.1: (a)Imagen (b)Trimapa de Lasso (c)Segmentación lograda con QMPF (Crédito: [50])

4.2. Textura

Las ideas implementadas en el algoritmo de segmentación asistida con trimapas y QMPF, que se plantean más adelante, se desarrollaron con el fin de incluir las texturas como información extra por lo tanto en esta sección se hablará sobre las texturas, su taxonomía y segmentación.

4.2.1. Introducción

Muchos algoritmos de visión máquina y procesamiento de imágenes hacen suposiciones simples sobre la uniformidad local de intensidades de la imagen. Sin embargo, con frecuencia imágenes de objetos reales no exhiben regiones de intensidades uniformes. Por ejemplo, la imagen de una superficie de madera no es uniforme pero contiene variaciones de intensidades que forman ciertos patrones repetidos llamados *textura visual*. Los patrones pueden ser el resultado de propiedades físicas de la superficie tales como rugosidad o filamentos orientados, que con frecuencia tienen una calidad táctil, o pueden ser el resultado de diferencias en la reflectancia debido al color en una superficie.
Se pueden reconocer texturas cuando se ven pero es muy dificil definirlas, tal vez sea por ello a las diversas definiciones de textura existentes en la literatura. Una definición de textura es: "Podemos observar una textura como aquello que consituye una región macroscópica. Su estructura es simplemente atribuida a patrones repetitivos

cuyos elementos son acomodados de acuerdo a una regla de posicionamiento" [56].
La textura en imágenes, definida como una función de variación espacial en las intensidades de los pixeles (valores de gris), es útil en una variedad de aplicaciones y ha sido objeto de estudio de muchos investigadores.

4.2.2. Una taxonomía de modelos de texturas

La textura es una propiedad de regiones; la textura de un punto es indefinida. Por lo tanto, la textura es una propiedad contextual y su definición involucra valores de gris en una vecindad espacial. El tamaño de la vecindad depende del tipo de textura, o el tamaño de los primitivos que definen la textura.
Si solamente unos pocos objetos primitivos están presentes, entonces un grupo contable de objetos es percibido en lugar de una imagen texturizada. En otras palabras, una textura es percibida cuando "formas" individuales significantes no están presentes.
Una imagen texturizada tiene un número de cualidades percibidas que juegan una rol importante al describir la textura. Laws [34] indentificó las siguientes propiedades que describen una textura: uniformidad, densidad, tosquedad, rugosidad, regularidad, linealidad, direccionalidad, dirección, frecuencia y fase. Algunas de estas cualidades no son independientes. Por ejemplo, la frecuencia no es independiente de la densidad y la propiedad de dirección aplica únicamente con texturas direccionales. El hecho de que la percepción de textura tiene tantas dimensiones diferentes es una razón importante por la cual no existe un método único de representación de textura.

Métodos estadísticos

Una de las cualidades que definen la textura es la distribución espacial de valores de gris. El uso de características estadísticas es uno de los primeros métodos propuestos en la literatura de visión máquina. En lo que sigue, se usará $I(x, y) : \{0 \leq x \leq N-1{,}0 \leq y \leq N-1\}$ para denotar una imagen de $N \times N$ con G niveles de gris. Un gran número de características de texturas ha sido propuesto.

Matrices de co-ocurrencia siglas en inglés GLCM. Estimaciones de co-ocurrencia espaciales estiman propiedades de una imagen relacionada con estadísticos de segundo orden. La $GLCM$ P_d $(G \times G)$ para un vector de desplazamiento $d = (dx, dy)$ está definida como sigue. La entrada (i, j) de P_d es el número de ocurrencias del par de niveles de gris i y j que se encuentran a una distancia d. Formalmente, está dada por

$$P_d(i, j) = |\{((r, s), (t, v)) : I(r, s) = i, I(t, v) = j\}| \tag{4.5}$$

donde $(r, s), (t, v) \in N \times N, (t, v) = (r + dx, s + dy)$. La matriz de co-ocurrencia es una matriz no simétrica. Pero una matriz de co-ocurrencia simétrica puede ser calculada por la fórmula $P = P_d + P_{-d}$. La matriz de co-ocurrencia revela ciertas propiedades acerca de la distribución espacial de los niveles de grises en una imagen con textura. Por ejemplo, si muchas de las entradas en la matriz de co-ocurrencia están concentradas en la diagonal, entonces la textura es rugosa con respecto al desplazamiento

d. Haralick [51] ha propuesto rasgos útiles de textura que pueden ser calculados a partir de la matriz de co-ocurrencia. Algunos de estos son: energía ($\sum_i \sum_j P_d^2(i,j)$), entropía ($-\sum_i \sum_j P_d(i,j) \log P_d(i,j)$), contraste ($\sum_i \sum_j (i-j)^2 P_d(i,j)$), homogeneidad ($\sum_i \sum_j \frac{P_d(i,j)}{1+|i-j|}$) y correlación ($\frac{\sum_i \sum_j (i-\mu_x)(j-\mu_y) P_d(i,j)}{\sigma_x \sigma_y}$) (ver (fig 4.2)).
Donde μ_x y μ_y son las medias y σ_x y σ_y son las desviaciones estandar de $P_d(x)$ y $P_d(y)$, respectivamente, donde $P_d(x) = \sum_j P_d(x,j)$ y $P_d(y) = \sum_i P_d(i,y)$.
La matriz de co-ocurrencia sufre un número de dificultades. No existe un buen mé-

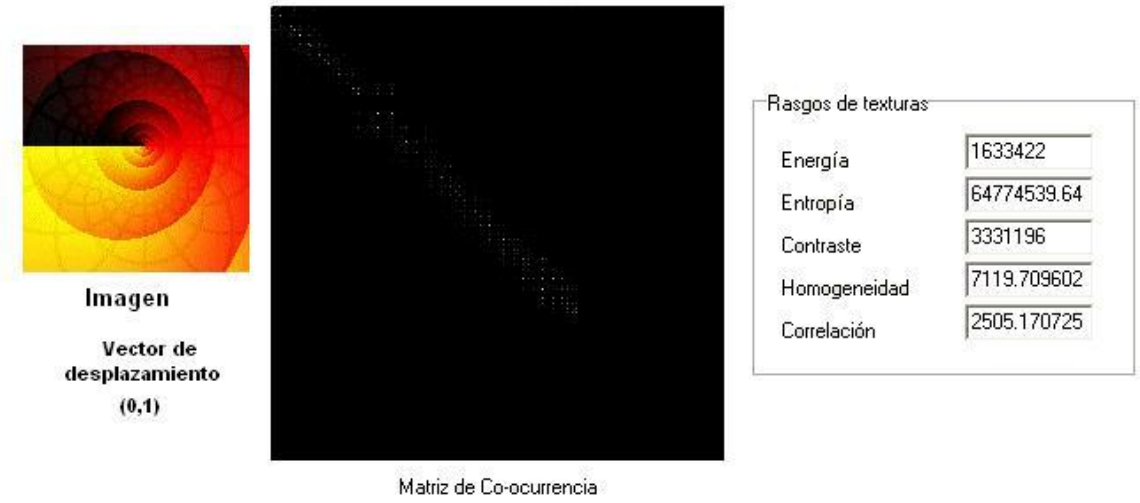

Figura 4.2: Rasgos de Haralick

todo establecido de seleccionar el vector de desplazamiento d y calcular las matrices de co-ocurrencia para distintos valores de d no siempre es factible. Para una d dada, pueden calcularse un gran número de rasgos a partir de la matriz de co-ocurrencia. Esto significa que debe realizarse una selección de los rasgos más relevantes. Es por esto que la matriz de co-ocurrencia ha sido utilizada principalmente en clasificación y no en segmentación.

Rasgos de autocorrelación. Una importante propiedad de muchas texturas es la naturaleza repetitiva de los elementos de textura en la imagen. La función de autocorrelación de una imagen puede ser usada para identificar la regularidad al igual que la fineza/rugosidad de la textura presente en la imagen. Formalmente, la función de autocorrelación de una imagen está definida por:

$$\rho(x,y) = \frac{\sum_{u=0}^{N} \sum_{v=0}^{N} I(u,v) I(u+x, v+y)}{\sum_{u=0}^{N} \sum_{v=0}^{N} I^2(u,v)} \tag{4.6}$$

Esta función está relacionada con el tamaño del primitivo de la textura. Si la textura es rugosa, entonces la autocorrelación disminuirá lentamente; de lo contrario, lo hará rápidamente. La función de autocorrelación está relacionada con el espectro de poder

de la transformada de Fourier.

Métodos geométricos

El método de análisis usalmente depende de las propiedades geométricas de los elementos de textura o primitivos. Una vez que los elementos de textura son identificados en la imagen, existen 2 aproximaciones mayores para analizar la textura. Una es calcular las propiedades estadísticas de los elementos de textura extraídos y utilizarlos como rasgos de textura. La otra intenta extraer las reglas de ubicación, con métodos geométricos y sintácticos,que describen la textura.

Rasgos Voronoi tessellation. La teselación de Voronoi fué propuesta en [60] debido a sus propiedades deseables en definir vecinos locales espaciales y porque las distribuciones locales espaciales de los tokens[1] se reflejan en las formas de los polígonos Voronoi. Primero, los tokens de textura son extraidos y despues la teselación es construida. Rasgos de cada célula Voronoi son extraidos y los tokens con características similares son agrupadas para construir regiones de textura uniformes.
Los rasgos de textura basados en polígonos Voronoi han sido usados para la segmentación de imágenes texturizadas. El algoritmo de segmentación está basado en fronteras, usando una comparación estadística de las colecciones de tokens colindantes. Este algoritmo ha sido exitosamente utilizado para segmentar imágenes texturizadas de grises al igual que texturas sintéticas con estadísticos de segundo orden idénticos.

Métodos estructurales. Los modelos estructurales de textura asumen que las texturas están compuestas de texturas primitivas. La textura es producida por la ubicación de las primitivas de acuerdo a unas reglas de colocación. Esta clase de algoritmos, en general, está limitado en poder a menos que se trate con texturas muy regulares. El análisis estructural de texturas consiste de dos pasos principales:

1. Extracción de los elementos de textura.

2. Inferencia sobre la regla de colocación.

Métodos basados en modelos

Los métodos de análisis de textura basados en modelos se fundamentan en la construcción de un modelo de la imagen que puede utilizarse para describir textura y también para sintetizarla. Los parámetros del modelo capturan las cualidades de textura. Entre estos métodos se encuentran los modelos de campos aleatorios Markovianos y los fractales.

[1] Los tokens pueden ser tan simples como puntos con un gradiente alto en la imagen o estructuras complejas tales como segmentos de líneas o fronteras cerradas.

Métodos de procesamiento de señales

La investigación en psicofísica ha proporcionado evidencia de que el cerebro humano hace un análisis de frecuencia de la imagen. La textura se ajusta a este tipo de análisis debido a su estructura. Muchos métodos tratan de calcular propiedades a través de imágenes filtradas para tareas de clasificación y segmentación.

Filtros de dominio espacial. Los filtros de dominio espacial es el método más directo de capturar las propiedades de textura de una imagen. Las últimas pruebas para definir tales métodos se concentran en la medida de la densidad de bordes por unidad de área. Texturas finas tienden a tener una mayor densidad de bordes por unidad de área que aquellas rugosas. La medida de rugosidad es usualmente calculada por máscaras de bordes tales como el operador Robert o el operador Laplaciano. Otros conjuntos de filtros espaciales están basados en momentos espaciales. Los momentos $(p+q)^{th}$ sobre una región R están dados por

$$m_{pq} = \sum_{(x,y)\in R} x^p y^q I(x,y)$$

Si la región r es un área rectangular local y los momentos son calculados alrededor de cada pixel en la imagen, entonces esto es equivalente a filtrar la imagen por un conjunto de máscaras espaciales. Las imagenes filtradas que corresponden a los momentos son usados como rasgos de textura. Los rasgos basados en momentos han sido usados de forma exitosa en la segmentación de texturas [36] (ver fig (fig 4.3)).

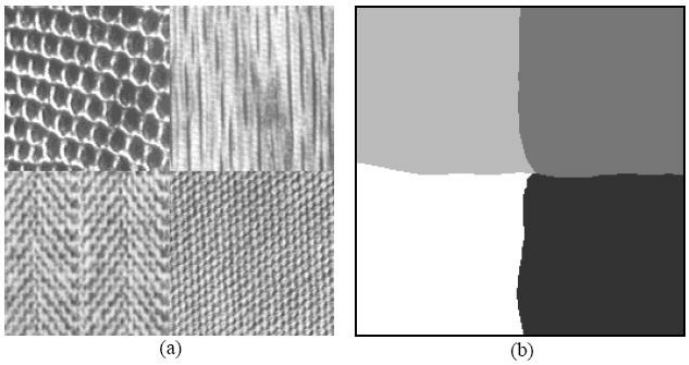

Figura 4.3: Ejemplo de segmentación de texturas basada en momentos (a)Imagen con cuatro regiones texturizadas (b)Segmentación final (Crédito: [36])

Filtros en el dominio de Fourier. El análisis de frecuencias de la imagen texturizada es mejor hacerlo en el dominio de Fourier. El filtrado en el Dominio de Fourier permite obtener rasgos de texturas. Han demostrado ser exitosos en la segmentación y clasificación de una gran variedad de imágnes naturales con estadísticos de segundo orden.

Modelos de Gabor y Wavelets. La transformada de Fourier es un análisis de la fre-

cuencia global contenida en una señal. Muchas aplicaciones requieren que el análisis sea localizado en el dominio espacial. Esto se maneja usualmente introduciendo una dependencia espacial en el análisis de Fourier. La forma clásica de hacer esto es a través de la llamada ventana de transformada de Fourier. La ventana de transformada de Fourier de una señal de una dimensión $f(x)$ está definida por

$$F_x(u,\xi) = \int_{-\infty}^{\infty} f(x)w(x-\xi)e^{-j2\pi ux}dx$$

Cuando la función ventana $w(x)$ es Gaussiana, la transformación se vuelve una transformación de Gabor. Los límites en la resolución del tiempo y el dominio de la frecuencia de la ventana de la transformada de Fourier están determinados por la desigualdad de incertidumbre de Heisenberg por:

$$\delta t \delta u \geq \frac{1}{4\pi}$$

Una vez que una ventana es elegida para la ventana de transformación de Fourier, la resolución tiempo-frecuencia se fija sobre todo el plano tiempo-frecuencia. Para vencer la limitación de resolución de la ventana de la transformación de Fourier, uno deja que δt y δu varíen en el dominio tiempo-frecuencia. Intuitivamente, la resolución del tiempo debe incrementarse cuando la frecuencia central del filtro analizado se incrementa. Esto se logra al usar una ventana cuyo ancho cambie conforme las frecuencias cambien.
Recuerde que una función $f(t)$ es escalada en el tiempo por a que es expresada como $f(at)$, la función se contrae si $a > 1$ y se expande cuando $a < 1$. Usando este hecho, la transformación wavelet puede escribirse como:

$$W_{f,a}(u,\xi) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} f(t)h * \left(\frac{t-\xi}{a}\right) dt$$

De esta forma, el impulso que responde del banco de filtros resulta una versión escalada de la misma función prototipo $h(t)$. Incluyendo en la ecuación anterior

$$h(t) = w(t)e^{-j2\pi ut}$$

se obtiene el modelo wavelet para el análisis de texturas. Usualmente el factor de escala a se basará en la frecuencia del filtro.
Los filtros de Gabor tiene propiedades de optimalidad deseables. Daugman [19] demostró que para funciones de Gabor de 2 dimensiones, la relaciones de incertidumbre $\delta x \delta u \geq \pi/4$ y $\delta y \delta v \geq \pi/4$ alcanzan el valor mínimo. Aquí δx y δy son anchos efectivos en el dominio espacial y δu y δv son anchos de banda efctivos en el dominio de frecuencia.
Una función de Gabor de 2 dimensiones consiste de una onda plana sinosoidal de cierta frecuencia y orientación modulado por una Gaussiana envolvente:

$$f(x,y) = exp\left(-\frac{1}{2}\left[\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}\right]\right)\cos(2\pi u_0 x + \phi)$$

donde u_0 y ϕ son la frecuencia y fase de la onda sinosoidal. Los valores σ_x y σ_y son los tamaños de la Gaussiana envolvente en las direcciones x y y, respectivamente. La función Gabor en una orientación arbitraria θ_0 puede obtenerse de la ecuación anterior con una rotación rígida del plano $x - y$ por θ_0. El filtro de Gabor es un filtro selectivo de frecuencia y orientación.
Rasgos de textura usados en filtros de Gabor han sido usados en tareas de segmentación y clasificación de texturas de forma exitosa.

4.2.3. Segmentación de texturas

La segmentación de texturas es un problema difícil porque uno usalmente no sabe *a priori* que tipo de texturas existen en una imagen, cuantas texturas diferentes existen, y que regiones en la imagen tienen cada textura. De hecho, uno no necesita saber que texturas en específico existen en la imagen para realizar la segmentación. Todo lo que se necesita es una forma de decir que dos texturas (usualmente en regiones adyacentes) son diferentes.
Las dos corrientes generales para la segmentación de textura son **análogas** a los métodos para segmentar imágenes: los basados en regiones y los basados en fronteras. En la aproximación por regiones, un trata de identificar regiones de la imagen que tienen una textura uniforme. Pixeles o regiones pequeñas locales emergen basadas en la similaridad de una propiedad de textura. Este método tiene la ventaja de que las fronteras de regiones son siempre cerradas y por lo tanto, las regiones con diferentes texturas son bien separadas. Tiene la desventaja, sin embargo, que en muchos métodos de segmentación basados en regiones uno debe especificar de antemano el número de diferentes texturas presentes en la imagen.
Las aproximaciones basadas en fronteras están basadas en la detección de diferencias en textura de regiones adyacentes. Entonces las fronteras son detectadas donde existen diferencias en textura. En este método, uno no necesita conocer el número de regiones texturizadas en la imagen antes de comenzar la segmentación. Sin embargo, las fronteras pueden tener huecos y 2 regiones con diferentes texturas pueden no ser identificadas como regiones cerradas separadas. Estrictamente, hablando, los métodos basados en frontera logran la segmentación solamente si todas las fronteras detectaron las curvas cerradas.
La segmentación basada en fronteras, detecta las fronteras tomando dos ventanas adyacentes y decidiendo si las texturas en las dos ventanas corresponden a la misma textura o a diferentes. Si se deciden que las dos texturas son diferentes, el punto es marcado como pixel frontera. Jain y Farrokhnia [32] dieron un ejemplo de integrar los métodos basados en regiones con los basados en fronteras para obtener un método de segmentación más limpio y robusto. Ellos usaron los rasgos de textura calculados de un banco de filtros de Gabor para llevar a cabo la segmentación basada en regiones (ver fig 4.4).

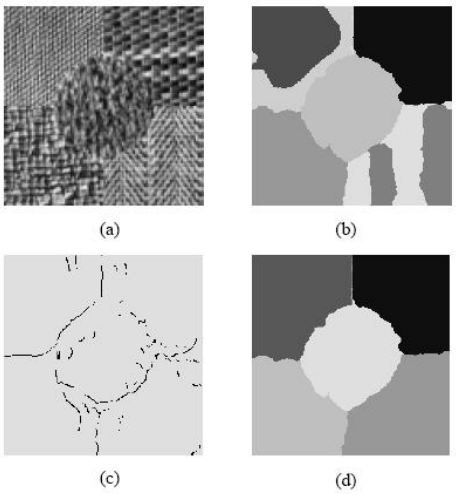

Figura 4.4: Ejemplo de segmentación de texturas integrando procesos basado en regiones y basado en bordes (a)Imagen con cinco texturas naturales (b)Siete regiones encontradas por segmentación basada en regiones (c)Resultado obtenido por la segmentación basada en bordes (d)Segmentación final al integrar las segmentaciones anteriores (Crédito: [32])

4.2.4. Segmentación de imágenes con color y textura

La textura y el color son dos rasgos importantes que pueden ser utilizados para detectar variaciones espaciales en una imagen. La inclusión de rasgos de textura en la segmentación de imágenes resulta complicado debido a que no existe una definición global de lo que es una textura y por lo mismo no existen rasgos universales para estas. Es hasta finales del siglo XX cuando el estudio de la segmentación de ambos rasgos se unen para buscar alcanzar una mayor precisión en los resultados.
Es en 1998 cuando Serge Belongie *et. al* [24] presentan por primera vez un algoritmo de segmentación de imágenes en general basado en color y textura. En donde además de los canales de color agregan otros tres rasgos de texturas que son polaridad, anisotropía y contraste. La elección de estos rasgos al parecer es porque que les resultan convenientes, sin mayor justificación de porque eligieron esos rasgos y no otros.
En el 2002 Junqing Chen *et al.* [33] proponen realizar primero la segmentación basada en rasgos de textura y posteriormente la segmentación basada en color por características para refinarla. La segmentación de la textura la convierten en un problema de clasificación al suponer únicamente cuatro clases de textura que la llevan a cabo a través wavelets. Las imágenes con las que trabajaron fueron principalmente de paisajes obteniendo buenos resultados, aunque limitados.
Tres años más tarde Mustafa Ösden *et al.* [64] utilizan la misma idea que en [33] para obtener la segmentación de textura pero para agregarla como información adicional además de los rasgos espaciales y color en el algoritmo *mean shift* para crear un espacio de características de dimensión mayor y buscar mejorar la segmentación.
En fin a partir del 2000 se han presentado una gran variedad de trabajos de segmentación donde incluyen la textura, cada uno utilizando distintos rasgos que funcionan

adecuadamente bajo ciertas condiciones. Es en el 2006 cuando se presenta un algoritmo de segmentación interactiva utilizando información de textura y color [20] que contiene algo innovador que es tratar de representar las texturas de una imagen sin la necesidad de recurrir al estado del arte. Lo que hacen es pasar una ventana, de tamaño previamente fijado, por toda la imagen y simplemente extraer la media y asignarla al pixel centrado. Al ser dificil poder elegir una forma de extraer la textura, resulta interesante como evitan la elección al aplicar este sencillo filtro. Lo que utilizan son campos aleatorios Markovianos donde los *matting factors* se obtienen con máquinas de soporte vectorial e incluyen además al igual que los QMPF información de los bordes.

4.3. Histogramas vs GMMs

En el trabajo [50], en el que se basa esta tesis, utilizan los histogramas conjuntos de los canales de color, obtenidos a partir de un trimapa, para evaluar cada pixel $x \in \mathcal{U}$. Los histogramas son una herramienta muy poderosa para calcular densidades de forma no paramétrica. Una desventaja de estos es que se requiere bastante memoria para almacenar un histograma conjunto de los canales de color que definitivamente sería impráctico si se incrementa la dimensión del problema, además de que se complica su manipulación. Esto último no nos favorecía debido a que entre las ideas propuestas se encontraba la de agregar rasgos de textura.
Una de las alternativas aquí explorada es utilizar modelos de mezclas de Gaussianas (GMMs) para calcular las densidades multimodales de cada región k proporcionada por el trimapa. Estos son un tipo de modelo de densidad que compromete un número distribuciones Gaussianas que combinadas proveen de una densidad multimodal.
Los GMMs han mostrado proveer gran flexibilidad y precisión al modelar una muestra de datos, dado que son capaces de suavizar regiones entre datos esparcidos y proveen restricciones más ajustadas al calcular verosimilitudes. De hecho en [24] se realiza una comparación entre el uso de histogramas y GMMs, favoreciendo en la mayoría de los casos prueba el uso de GMMs.
Para calcular los parámetros de las M Gaussianas que conforman el GMM se utiliza el algoritmo *Esperanza-Maximización* que al modificarse a través del estimador de *Mínima Longitud de Descripción* (MDL) se vuelve una opción no supervisada al igual que los histogramas.

4.3.1. Algoritmo Esperanza-Maximización

El problema de estimar una función densidad puede definirse como: dado un conjunto N de puntos en D dimensiones, $x_1, \cdots, x_N \in \mathbb{R}^D$, y una familia $\mathcal{F}$ de funciones de densidad de probabilidad, encontrar la función de densidad $f(x) \in \mathcal{F}$ que sea más probable de haber generado los puntos dados. El algoritmo *Esperanza Maximización* (EM) [10] [58] [11] es un procedimiento iterativo que resuelve el problema de estimar la densidad a través del cálculo de la Máxima Verosimilitud, cuando existen datos faltantes, con el fin de estimar los parámetros de un modelo para el cual ciertos datos

se ajustan de la mejor manera posible.
En cada iteración del EM se llevan a cabo dos pasos: El paso-E, y el paso-M. En la expectación, paso-E, los datos faltantes son estimados a partir de los datos observados y la estimación actual de los parámetros del modelo. En el paso-M, la función de verosimilitud se maximiza bajo la suposición de que los datos faltantes son conocidos. La convergencia se asegura dado que el algoritmo garantiza el incremento de la verosimilitud en cada iteración.

Máxima Verosimilitud

Se tiene una función de densidad $p(x|\Theta)$ que es gobernada por el conjunto de parámetros Θ y un conjunto de datos $\mathcal{X}$ de tamaño N. Se asume que los datos son i.i.d.. Por lo tanto la densidad para está muestra es

$$p(\mathcal{X}|\Theta) = \prod_{i=1}^{N} p(x_i|\Theta) = \mathcal{L}(\Theta|\mathcal{X}).$$

donde $\mathcal{L}(\Theta|\mathcal{X})$ es conocida como la verosimilitud de los parámetros dados los datos. La verosimilitud se observa como una función de los parámetros Θ donde los datos $\mathcal{X}$ son dados. En el problema de la Máxima Verosimilitud (ML), el objetivo es encontrar la Θ que maximice $\mathcal{L}$. Es decir, se busca encontrar Θ^* tal que

$$\Theta^* = \arg\max_{\Theta} \mathcal{L}(\Theta|\mathcal{X}).$$

Con frecuencia no es posible resolver el problema de forma analítica y debe resolverse con técnicas más elaboradas.

Algoritmo general

El algoritmo EM es un método que nos permite resolver el problema MV cuando existen datos faltantes y también cuando no es posible resolverlo de forma analítica pero cuando la función de verosimilitud puede simplificarse asumiendo la existencia de la solución y los valores de parámetros adicionales pero faltantes (escondidos). Los datos $\mathcal{X}$ son llamados *datos incompletos* y se asume que fueron generados por una misma distribución. Se presume que el conjunto de datos completo existe $\mathcal{Z} = (\mathcal{X}, \mathcal{Y})$ y la función de densidad conjunta es:

$$p(z|\Theta) = p(x, y|\Theta) = p(y|x, \Theta)p(x|\Theta)$$

Con esta nueva función de densidad, se puede definir una nueva función de verosimilitud, $\mathcal{L}(\Theta|\mathcal{Z}) = \mathcal{L}(\Theta|\mathcal{X}, \mathcal{Y}) = p(\mathcal{X}, \mathcal{Y}|\Theta)$, conocida como verosimilitud de datos completos. Cabe notar que esta función es de hecho una variable aleatoria dado que la información faltante $\mathcal{Y}$ es desconocida. Se puede pensar que $\mathcal{L}(\Theta|\mathcal{X}, \mathcal{Y}) = h_{\mathcal{X},\Theta}(\mathcal{Y})$ para alguna función $h_{\mathcal{X},\Theta}(\cdot)$ donde $\mathcal{X}$ y Θ son constantes y $\mathcal{Y}$ es una variable aleatoria. Entonces la verosimilitud $\mathcal{L}(\Theta|\mathcal{X})$ se nombrará como la verosimilitud de datos incompletos.

El algoritmo EM encuentra primero el valor esperado de $\log p(\mathcal{X}, \mathcal{Y}|\Theta)$ con respecto a los datos faltantes $\mathcal{Y}$ dado los datos observados $\mathcal{X}$ y la estimación actual de los parámetros. Es decir, se define:

$$Q(\Theta, \Theta^{(i-1)}) = E\left[\log p(\mathcal{X}, \mathcal{Y}|\Theta)|\mathcal{X}, \Theta^{(i-1)}\right] \tag{4.7}$$

donde $\Theta^{(i-1)}$ son los actuales parámetros estimados que se usan para evaluar la esperanza y Θ son los nuevos parámetros que se optimizan para incrementar Q. En (4.7) $\mathcal{X}$ y $\Theta^{(i-1)}$ son constantes, Θ es una variable normal que se desea ajustar, y $\mathcal{Y}$ es una variable aleatoria con distribución $f(y|\mathcal{X}, \Theta^{(i-1)})$. Entonces la ecuación (4.7) puede reescribirse como:

$$E[\log p(\mathcal{X}, \mathcal{Y}|\Theta)|\mathcal{X}, \Theta^{(i-1)}] = \int_{y\in\Upsilon} \log p(\mathcal{X}, y|\Theta) f(y|\mathcal{X}, \Theta^{(i-1)}) dy. \tag{4.8}$$

donde $f(y|\mathcal{X}, \Theta^{(i-1)})$ es una distribución marginal de los datos faltantes que depende de los datos observados $\mathcal{X}$ y las estimaciones actuales de los parámetros, y Υ es el espacio de valores de y.
La evaluación de la esperanza es el paso-E del algoritmo. Note el significado de los dos argumentos en la función $Q(\Theta, \Theta')$. El primero argumento Θ corresponde a los parámetros que serán optimizados para maximizar la verosimilitud y el segundo Θ' corresponde a los parámetros que se usarán para evaluar la esperanza.
El segundo paso (paso-M) del algoritmo EM es para maximizar la esperanza calculada en el primer paso. Es encontrar:

$$\Theta^{(i)} = \arg\max_{\Theta} Q(\Theta, \Theta^{(i-1)}).$$

Los pasos anteriores se repiten cuantas veces sea necesario. Cada iteración garantiza el incremento de la log-verosimilitud y el algoritmo garantiza converger a un máximo local.

EM para una mezcla de distribuciones

El problema de estimar los parámetros de una mezcla de densidades es una de las mayores aplicaciones del algoritmo EM en la comunidad de reconocimiento de patrones. En este caso se asume al modelo probabilístico como:

$$p(x|\Theta) = \sum_{i=1}^{M} \alpha_i p(x|\theta_i)$$

donde los parámetros son $\Theta = (\alpha_1, \cdots, \alpha_M, \theta_1, \cdots, \theta_M)$ tal que $\sum_{i=1}^{M} \alpha_i = 1$ y cada p es una función de densidad parametrizada por θ_i. En otras palabras, se tiene M densidades mezcladas con M coeficientes α_i.
La log-verosimilitud de datos incompletos para esta densidad de los datos $\mathcal{X}$ está dada por:

$$log(\mathcal{L}(\Theta|\mathcal{X})) = \log \prod_{i=1}^{N} p(x_i|\Theta) = \sum_{i=1}^{N} log\left(\sum_{j=1}^{M} \alpha_j p(x_i|\theta_j)\right)$$

lo cual es difícil de optimizar porque contiene el logaritmo de la suma. Si se considera $\mathcal{X}$ como incompleto, y se asume la existencia de datos no observados $\mathcal{Y} = \{y_i\}_{i=1}^N$ cuyos valores nos informan que densidad de la mezcla generó cada dato, la verosimilitud se simplifica significativamente. Es decir, se asume que $y_i \in 1, \cdots, M$ para cada i, y $y_i = k$ si la i-ésima muestra fue generada por el k-ésimo componente de la mezcla. Si se conocieran los valores de $\mathcal{Y}$, y asumiendo una distribución uniforme para Θ se tiene

$$\mathcal{L}(\Theta|\mathcal{X}, \mathcal{Y}) = P(\mathcal{X}, \mathcal{Y}|\Theta)P(\Theta) = P(\mathcal{X}, \mathcal{Y}|\Theta)$$

Entonces la verosimilitud sería:

$$\log(\mathcal{L}(\Theta|\mathcal{X}, \mathcal{Y})) = \log(P(\mathcal{X}, \mathcal{Y}|\Theta)) = \sum_{i=1}^{N} \log(P(x_i|y_i)P(y)) = \sum_{i=1}^{N} \log(\alpha_{y_i} p(x_i|\theta_{y_i}))$$

lo cual nos da una forma particular de cada densidad de la mezcla y puede ser optimizado usando una gran variedad de técnicas.

El problema, en efecto, es que no se conocen los valores de $\mathcal{Y}$. Pero si se asume que $\mathcal{Y}$ es un vector aleatorio entonces se puede proceder.

Primero se debe derivar una expresión de la distribución de los datos no observados. Por lo que deben inicializarse los parámetros de la mezcla de densidades, i.e., se asume que $\Theta^g = (\alpha_i^g, \cdots, \alpha_M^g, \theta_1^g, \cdots, \theta_M^g)$ son los parámetros apropiados de la verosimilitud $\mathcal{L}(\Theta^g|\mathcal{X}, \mathcal{Y})$. Dado Θ^g, se puede fácilmente calcular $p(x_i|\theta_j^g)$ para cada i y j. Además α_j puede pensarse como probabilidades a priori de cada componente de la mezcla, esto es $\alpha_j = p(\text{componente } j)$. Por lo tanto usando la regla de Bayes[2], se puede calcular:

$$p(y_i|x_i, \Theta^g) = \frac{\alpha_{y_i}^g p_{y_i}(x_i|\theta_{y_i}^g)}{p(x_i|\Theta^g)} = \frac{\alpha_{y_i}^g p_{y_i}(x_i|\theta_{y_i}^g)}{\sum_{k=1}^{M} \alpha_k^g p_k(x_i|\theta_k^g)}$$

y asumiendo independencia

$$p(y|\mathcal{X}, \Theta^g) = \prod_{i=1}^{N} p(y_i|x_i, \Theta^g)$$

Al observar de nuevo la ecuación (4.8), se nota que en este caso se ha obtenido la densidad marginal deseada al asumir la existencia de variables ocultas y valores iniciales para los parámetros de su distribución.

En este caso, la ecuación (4.7) toma la forma:

$$Q(\Theta, \Theta^g) = \sum_{y \in \Upsilon} \log(\mathcal{L}(\Theta|\mathcal{X}, y)) p(y|\mathcal{X}, \Theta^g)$$

y con cierta manipulación algebraica se tiene

$$Q(\Theta, \Theta^g) = \sum_{l=1}^{M} \sum_{i=1}^{N} \log(\alpha_l) p(l|x_i, \Theta^g) + \sum_{l=1}^{M} \sum_{i=1}^{N} \log(p_l(x_i|\theta_l)) p(l|x_i, \Theta^g) \tag{4.9}$$

[2]La regla de Bayes dice que para un espacio de eventos S con k particiones $A_1, A_2, \cdots, A_k$ y sea B un evento cualquiera de S, entonces la probabilidad condicional de que se de una partición A_i dado que se dió el evento B se calcula como $P(A_i|B) = \frac{P(A_i)P(B|A_i)}{\sum_{i=1}^{k} P(A_i)P(B|A_i)}$

Para maximizar esta expresión, se puede maximizar el término que contiene α_l y posteriormente el término que contiene θ_l de forma independiente dado que no están relacionados.
Si maximizamos (4.9)con repecto a α_l obtenemos:

$$\alpha_l^{\text{nuevo}} = \frac{1}{N}\sum_{i=1}^{N} p(l|x_i, \Theta^g)$$

Si la mezcla de densidades es de Gaussianas de dimensión d entonces los parámetros θ_l de cada una son su media μ y su matriz de covarianza Σ. La función de densidad de cada Gaussiana está dada por

$$p_l(x|\mu_l, \Sigma_l) = \frac{1}{(2\pi)^{d/2}\,|\Sigma_l|^{1/2}} e^{-\frac{1}{2}(x-\mu_l)^T \Sigma_l^{-1}(x-\mu_l)}.$$

Maximizando (4.9) con respecto a μ y Σ se obtienen las siguiente fórmulas:

$$\mu_l^{\text{nuevo}} = \frac{\sum_{i=1}^{N} x_i p(l|x_i, \Theta^g)}{\sum_{i=1}^{N} p(l|x_i, \Theta^g)}$$

$$\Sigma_l^{\text{nuevo}} = \frac{\sum_{i=1}^{N} p(l|x_i, \Theta^g)(x_i - \mu_l^{\text{nuevo}})(x_i - \mu_l^{\text{nuevo}})^T}{\sum_{i=1}^{N} p(l|x_i, \Theta^g)}$$

Finalmente el algoritmo EM es de la siguiente forma:

1. Se inicializan los parámetros $p_l^{(0)}$,$m_l^{(0)}$ y $\Sigma_l^{(0)}$.

2. Se realizan los siguiente cálculos, en el orden presentados, hasta la convergencia.

   Paso-E

   $$p(l|x_i, \Theta^{(t)}) = \frac{\alpha_l^{(t)} p_l(x_i|\Theta^{(t)})}{\sum_{k=1}^{M} \alpha_k^{(t)} p_k(x_i|\Theta^{(t)})}$$

   Paso-M

   $$\alpha_l^{(t+1)} = \frac{1}{N}\sum_{n=1}^{N} p(l|x_n, \Theta^{(t)})$$

   $$\mu_l^{(t+1)} = \frac{\sum_{n=1}^{N} x_n p(l|x_n, \Theta^{(t)})}{\sum_{n=1}^{N} p(l|x_n, \Theta^{(t)})}$$

   $$\Sigma_l^{(t+1)} = \frac{\sum_{n=1}^{N} p(l|x_n, \Theta^{(t)})(x_n - \mu_l^{(t+1)})(x_n - \mu_l^{(t+1)})^T}{\sum_{n=1}^{N} p(l|x_n, \Theta^{(t)})}$$

4.3.2. Algoritmo Esperanza Maximización con Estimador de Mínima Longitud de Descripción

El principal conflicto del algoritmo EM es que se debe fijar M el número de Gaussianas, es por eso que buscamos modificar el EM para poder incluir a M dentro de los parámetros a estimar. Como ya se mencionó el estimador de máxima verosimilitud (ML) se utiliza para estimar los parámetros en el EM y está dado por

$$\Theta^*_{ML} = \arg\max_{\Theta} \log \prod_{i=1}^{N} p(x_i|M,\Theta)$$

Desafortunadamente, el estimador ML de M no está bien definido porque la verosimilitud siempre será mejor al elegir un número grande de Gaussianas. Intuitivamente, el logaritmo de la verosimilitud siempre se incrementará al incluir más Gaussianas dado que mientras más haya con mayor precisión se ajustarán los datos.
Este problema de estimar el orden de un modelo es conocido como identificación del orden, y ha sido estudiado por una gran variedad de investigadores. Los métodos para estimar el orden de un modelo tienden generalmente requerir la adición de un término de penalización en la log verosimiltud para evitar la sobreparametrización de modelos de órdenes alto. Una de las primeras aproximaciones para identificar el orden fué sugerido por Akaike [5], y requiere la minimización del llamado criterio de información AIC que está dado por:

$$AIC(K,\theta) = -2\log \prod_{i=1}^{N} p(x_i|M,\Theta) + 2L$$

donde L es la cantidad de parámetros requeridos para especificar el vector Θ. Para el caso de una mezcla de Gaussianas esta dado por

$$L = M\left(1 + Dim + \frac{(Dim+1)Dim}{2}\right) - 1$$

donde Dim es la dimensión de los datos originales $\mathcal{X}$.
Sin embargo, una desventaja importante del criterio AIC es que no es un estimador consistente, i.e. mientras el número de observaciones tiende a infinito, el valor estimado de M no converge al verdadero valor.
De forma alternativa, Rissanen propuso otro criterio que llamó Estimador de Mínima Longitud de Descripción (MDL). Este estimador funciona al buscar encontrar el orden del modelo que minimice el número de bits que serían requeridos para codificar tanto los datos muestreados x_n como el vector de parámetros Θ. Mientras que una implementación directa del estimador MDL depende en el método para codificar elegido, Rissanen desarrolló una expresión aproximada para la estimación basada en algunas suposiciones y la minimización de:

$$MDL(M,\Theta) = -\log \prod_{i=1}^{N} p(x_i|M,\Theta) + \frac{1}{2}L\log(NDim)$$

Note que la mayor diferencia entre los criterios AIC y MDL es la dependencia del término de penalización en el número total de datos evaluados $N \cdot \text{Dim}$. En la práctica, esto es importante dado que de otra forma más datos tenderían a que el modelo se sobreajustara. De hecho se ha demostrado que para una gran cantidad de problemas, el criterio MDL es un estimador consistente del orden del modelo. Desafortunadamente, la estimación del orden del modelo para una mezcla de modelos no cae en la clase problemas para los cuales el criterio MDL es consistente. Esto se debe a que la solución del problema de modelar con una mezcla siempre cae en una frontera del espacio restringido, por lo que los resultados en la distribución asintótica del MDL no son válidos. Un método alternativo para minimizar el criterio MDL está dado por

$$MDL(M, \Theta) = -\sum_{i=1}^{N} log\left(\sum_{j=1}^{M} \alpha_j p_j(x_i|\theta_j)\right) + \frac{1}{2} L \log(NDim) \tag{4.10}$$

Para derivar formalmente las ecuaciones con las cuales se actualizan los parámetros, primero debe calcularse la siguiente función

$$Q(\Theta; \Theta^{(i)}) = E\left[\log p(\mathcal{X}, \mathcal{Y}|\Theta)|\mathcal{X}, \Theta^{(i-1)}\right] - \frac{1}{2} L \log(NDim)$$

Un resultado fundamental del algoritmo EM es que para todo Θ se tiene que

$$MDL(K, \Theta) - MDL(K, \Theta^{(i)}) < Q(\Theta^{(i)}; \Theta^{(i)}) - Q(\Theta; \Theta^{(i)})$$

Este resultado es muy útil dado que para cualquier valor de Θ que incremente el valor de $Q(\Theta; \Theta^{(i)})$ garantiza la reducción del criterio MDL. El objetivo entonces del algoritmo EM sería realizar una optimización iterativa con respecto a Θ hasta que se alcance un mínimo local en la función MDL.
Para cambiar el orden del modelo M, se empezará con un número grande de clusters, y de forma secuencial se decrementará el valor de M. Para cada valor de M, se aplicará el algoritmo EM hasta converger a un mínimo local del funcional MDL. Después de haber hecho esto para cada valor de M, simplemente se seleccionará el valor de M y los parámetros correspondientes que coinciden con valor más pequeño del criterio MDL.
La forma en que se decrementa el número de clusters de M a $M-1$ es fusionando dos clusters para formar uno. Una vía efectiva para reducir el orden de un modelo es restringir que los parámetros de 2 clusters sean iguales. Por ejemplo, dos subclases, l y m, pueden fusionarse al restringir que sus parámetros de media y covarianza sean iguales.

$$\begin{aligned} \mu_l &= \mu_m = \mu_{(l,m)} \\ R_l &= R_m = R_{(l,m)} \end{aligned} \tag{4.11}$$

donde $\mu_{(l,m)}$ y $R_{(l,m)}$ denota la media y la covarianza del nuevo cluster, y se asume que los valores de α_l y α_m permanecen sin cambios para los 2 clusters que fueron fusionados. Se denota a este vector de parámetros por $\Theta_{(l,m)} \in \Omega^{(M)}$. Note que $\Theta_{(l,m)}$

especifica los parámetros de M clusters, por lo tanto es miembro de $\Omega^{(M)}$, pero que dos de estos (e.g. clusters l y m) tienen medias y covarianzas idénticas. De forma alternativa, se usa la notación $\Theta_{(l,m)^-} \in \Omega^{(M-1)}$ para denotar los parámetros de los $M-1$ clusters distintos en $\Theta_{(l,m)}$. Más aún, los dos clusters l y m se especifican como un sólo cluster (l,m) con media y covarianza dada en (4.11), con probabilidad dada por

$$\alpha_{(l,m)} = \alpha_l + \alpha_m \tag{4.12}$$

Usando las definiciones de $\Theta_{(l,m)}$, $\Theta_{(l,m)^-}$ y (4.10) se tiene que

$$MDL(M-1, \Theta_{(l,m)^-}) = MDL(M, \Theta_{(l,m)}) + \frac{1}{2}\left(1 + Dim + \frac{(Dim+1)Dim}{2}\right)\log(NDim)$$

El cambio en el criterio MDL está dado por

$$\begin{aligned}
& MDL(M-1, \Theta_{(l,m)^-}) - MDL(M, \Theta^{(i)}) \\
= \;& MDL(M-1, \Theta_{(l,m)^-}) - MDL(M, \Theta_{(l,m)}) + MDL(M, \Theta_{(l,m)}) - MDL(M, \Theta^{(i)}) \\
\leq \;& -\frac{1}{2}\left(1 + Dim + \frac{(Dim+1)Dim}{2}\right)\log(NDim) + Q(\Theta^{(i)}; \Theta^{(i)}) - Q(\Theta_{(l,m)}; \Theta^{(i)}) \\
\leq \;& -\frac{1}{2}\left(1 + Dim + \frac{(Dim+1)Dim}{2}\right)\log(NDim) \\
& + Q(\Theta^{(i)}; \Theta^{(i)}) - Q(\Theta^*; \Theta^{(i)}) + Q(\Theta^*; \Theta^{(i)}) - Q(\Theta^*_{(l,m)}; \Theta^{(i)})
\end{aligned}$$

donde Θ^* y $\Theta^*_{(l,m)}$ son los óptimos sin restricciones y con restricciones respectivamente. Se puede definir una función de distancia de la forma

$$d(l,m) = Q(\Theta^*; \Theta^{(i)}) - Q(\Theta^*_{(l,m)}; \Theta^{(i)}) \tag{4.13}$$

Esta función de distancia sirve como un límite superior en el cambio del criterio MDL

$$MDL(M-1, \Theta_{(l,m)^-}) - MDL(M, \Theta^{(i)}) \leq d(l,m) - \tfrac{1}{2}\left(1 + Dim + \tfrac{(Dim+1)Dim}{2}\right)\log(NDim) \tag{4.14}$$

A continuación se realizan algunas observaciones. El valor de $d(l,m)$ es siempre positivo debido a la forma de (4.13). De hecho, reducir el orden del modelo debería reducir la log verosimilitud de las observaciones dado que existen menos parámetros que ajustan los datos. En general, este incremento es compensado por el término del orden del modelo que es siempre negativo. sin embargo, dado que este término es indpendiente de la elección de l y m, no juega una papel esencial en seleccionar que clusters fusionar.

Con la función $d(l,m)$ definida, es ahora posible buscar sobre todo el conjunto de pares, (l,m), para encontrar el par de clusters que minimizan $d(l,m)$, que a su vez minimizan el límite superior en el cambio del criterior MDL

$$(l^*, m^*) = \arg\min_{(l,m)} d(l,m) \tag{4.15}$$

Aquellos clusters que cumplen la ecuación (4.15) son aquellos que deben fusionarse. El conjunto de parámetros $\Theta^*_{(l,m)}$ es usado como condición inicial del EM con $M-1$

clusters.
Antes de poder especificar el algoritmo final, se debe especificar la elección inicial de los parámetros $\Theta^{(1)}$ usada con un número grande de clusters. La elección de $\Theta^{(1)}$ es importante dado que el EM sólo garantiza la convergencia a un mínimo local. Los parámetros iniciales de cada cluster son:

$$\begin{aligned} \alpha_i^{(1)} &= \frac{1}{M_0} \\ \mu_i^{(1)} &= x_n \text{ donde } n = \lfloor (i-1)(N-1)/(M_0-1) \rfloor + 1 \\ \Sigma_i^{(1)} &= \frac{1}{N} \sum_{n=1}^{N} x_n x_n^t \end{aligned} \tag{4.16}$$

donde M_0 es el número inicial de clusters.
Finalmente el algoritmo EM-MDL queda como sigue:

1. Elegir un número inicial de clusters, M_0, grande.
2. Inicializar $\Theta^{(1)}$ usando (4.16).
3. Aplicar el algoritmo EM hasta que el cambio en $MDL(M, \theta)$ sea menor que un ϵ.
4. Guardar el parámetro $\Theta^{(M,\text{final})}$ y el valor $MDL(M, \Theta^{(M,\text{final})})$.
5. Si el número de clusters es mayor que uno, aplicar (4.15) para reducir el número de clusters con los pares adecuados, hacer $M \leftarrow M - 1$ y regresar al paso 3.
6. Elegir el valor de M^* y los parámetros $\Theta^{(M^*,\text{final})}$ que minimizan el valor de MDL.

Este último algoritmo fue implementado por un grupo de la Universidad de Purdue [22] y fue el utilizado en este trabajo para modelar cada región k señalada por el trimapa.

4.3.3. Verosimilitud a partir de un GMM

Para calcular la verosimilitud del pixel $x \in \mathcal{U}$ con respecto al k-ésimo GMM sólo tiene que evaluarse como

$$v_k(x) = \sum_{i=1}^{M} \alpha_i p_i(x|\theta_i) \tag{4.17}$$

y posteriormente normalizarse. Los resultados en un principio fueron muy alentadores debido a que estos casi segmentaban la imagen como se muestra en (fig 4.5) pero lamentablemente la segmentación no era tan buena para no requerir el auxilio de un post-proceso como QMPF. Además de que la segmentación obtenida era casi dura provocando que el QMPF no pudiera corregir los "matting factors" y por lo tanto se estanca fácilmente en un mínimo local. Se optó por sumarle una matriz identidad

Figura 4.5: Verosimilitud obtenida del GMM del trimapa de la (fig 2.1)

multiplicada por un factor ω a la matriz de covarianza de cada distribución normal multivariada de todas las GMMs:

$$\tilde{\Sigma}_l = \Sigma_l + \omega I.$$

Esto para cubrir áreas que anteriormente no habían sido alcanzadas por la verosimilitud. La elección del valor de ω se realizó haciendo pruebas a una muestra de imágenes y la opción que se consideró adecuada fue $\omega = 3$ (ver (fig 4.7)). Ahora para evitar la segmentación casi dura obtenida por las verosimilitudes se creó un umbral h que debían superar las k verosimilitudes obtenidas de tal forma que al menos una sea significativa. En caso de no hacerlo la verosimilitud de cada región k se calcularía como $1/K$ en lugar de llevar a cabo la normalización (ver (fig 4.6)). De igual forma que para el caso de ω se realizaron pruebas a un grupo de imágenes y el valor del umbral se fijó a $h = 1e^{-6}$ (ver (fig 4.8)).

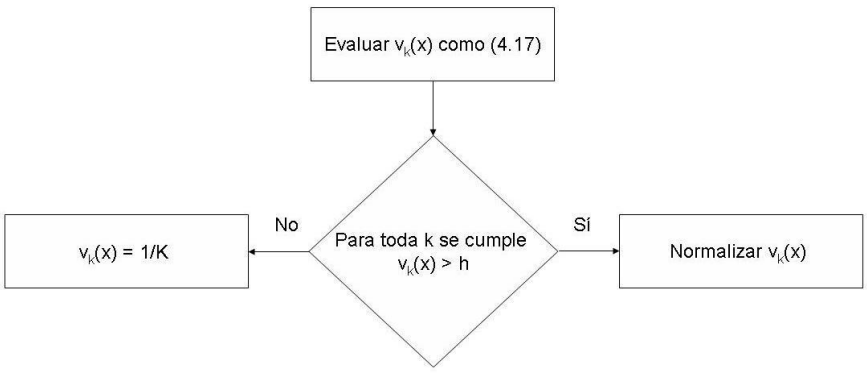

Figura 4.6: Diagrama de flujo sobre la acción del umbral sobre las verosimilitudes

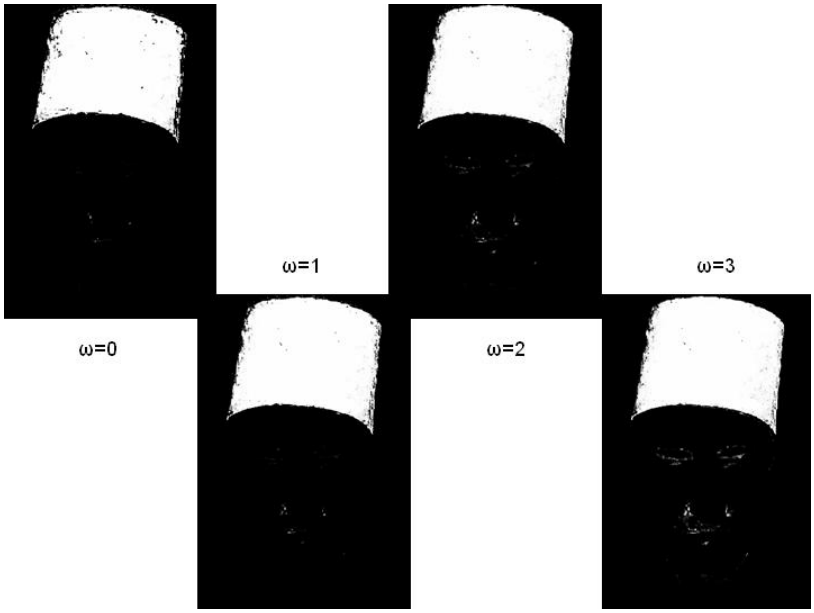

Figura 4.7: Verosimilitudes distintas modificando al factor ω del trimapa de (fig 2.1)

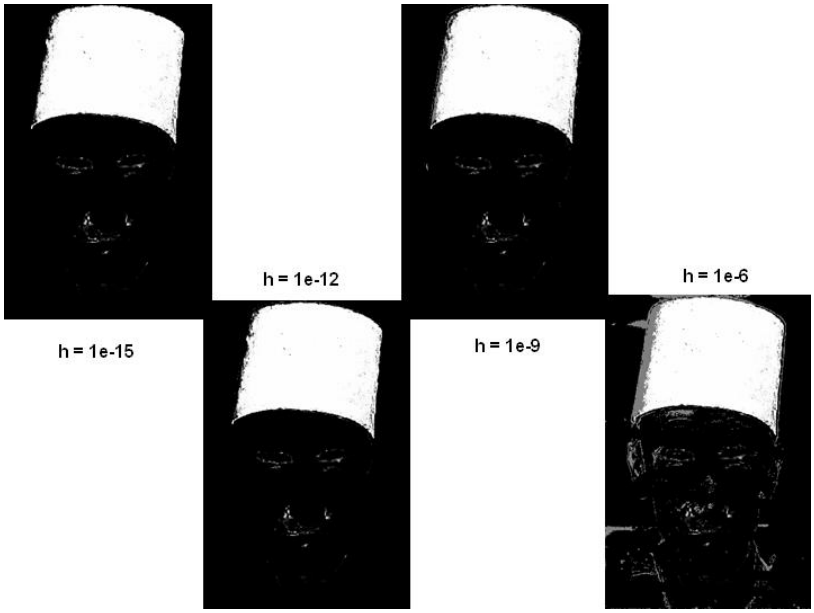

Figura 4.8: Verosimilitudes distintas con $\omega = 3$ y modificando al factor h del trimapa de (fig 2.1)

4.4. Segmentación robusta ante imágenes con textura

Para trabajar la robustez ante texturas se propuso en un principio buscar rasgos adecuados de estas para agregar como características extra de la imagen sin embargo la decisión de discriminar unos rasgos con respecto a otros resultaba muy complicada porque, como se vió en la sección sobre texturas, todos estos tienen sus limitantes además de que fueron desarrollados en un contexto exclusivo sobre texturas.
Posteriormente se pensó, para evitar la selección de rasgos, que cada pixel $x \in \mathcal{U}$ fuera representado a través de una distribución Gaussiana. Esto se haría pasando una ventana de tamaño fijo a través de cada pixel y los n pixeles contenidos en esta deberían de utilizarse para calcular los parámetros μ y σ de la Gaussiana, como

$$\mu = \frac{\sum_{i=1}^{n} x_i}{n} \qquad \sigma = \frac{\sum_{i=1}^{n} (x_i - \mu)^2}{n-1}$$

Una vez obtenida una Gaussiana por cada pixel se debería definir una métrica que se utilizaría para obtener la distancia entre cada Gaussiana y los GMMs obtenidos de cada región k obtenidos por el algoritmo EM MDL. Esta distancia sustituiría a $-\log v_k(x, \theta)$ en (3.17).
Para dos distribuciones p y q, la distancia $d(p, q)$

- Satisface la reflexividad $d(p, p) = 0$ y por lo tanto, en la mayoría de los casos, el aislamiento $d(p, q) = 0 \Rightarrow p = q$.
- Ocasionalmente es simétrica $d(p, q) = d(q, p)$.
- Aunque difícilmente satisface la desigualdad del triángulo:

  $$d(p, q) + d(q, r) \geq d(p, r)$$

  siempre satisface la versión relajada

  $$d(p, q) + d(q, r) \geq c \cdot d(p, r) \qquad c < 1$$

Existen muchos tipos de métricas entre las cuales encontramos:

- Distancia Kullback-Leibler

  $$KL(p, q) = \sum_i p_i \log \frac{p_i}{q_i}$$

- Distancia Jensen-Shannon

  $$JS_{\alpha,\beta}(p, q) = \alpha KL(p, m) + \beta KL(q, m), \qquad m = \alpha p + \beta q, \qquad \alpha + \beta = 1$$

- Distancia $\mathcal{X}^2$

  $$\mathcal{X}^2(p, q) = \sum_i \frac{(p_i - q_i)^2}{q_i}$$

- Distancia Δ

$$\Delta(p,q) = \sum_i \frac{(p_i - q_i)^2}{p_i + q_i}$$

donde p y q son dos distribuciones diferentes y p_i y q_i representan muestras de cada distribución. Existen otras distancias creadas por la comunidad de visión como la *Earth Mover's Distance* EMD [63] y la métrica de similaridad [16], sin embargo los computólogos al momento de requerir de un medida entre distribuciones recurrían comunmente a la distancia Kullback-Leibler (KL).
Lamentablemente para calcular la distancia KL solamente existe fórmula cerrada cuando se trata de dos distribuciones normales multivariadas. Por lo tanto se tenía que hacer a través de simulaciones de Monte-Carlo, i.e. se muestrea la distribución Gaussiana de cada pixel y se evalúa con respecto a cada GMM. El muestrear de la distribución Gaussiana además de requerir menos cálculos se tenía que hacer porque la distancia KL no es simétrica, i.e. $KL(p,q) \neq KL(q,p)$[3] y el actuar de esta forma permitiría muestrear una sola ocasión. Aun así esto resultaba poco alentador debido a que tardaría mucho tiempo en efectuarse, dado que tendrían que muestrearse todas las Gaussians de los pixeles no clasificados y no se cumpliría con la rapidez que se mencionó entre las propiedades deseadas de la segmentación asistida.
En [31] se presentan dos alternativas a las simulaciones de Monte-Carlo para evaluar la distancia KL. Buscan la distancia entre dos GMMs y asumen como verdadera la distancia KL obtenida a partir de la simulación de Monte-Carlo con 500 datos muestreados. Sus resultados, aunque mejoran indiscutiblemente el tiempo de procesamiento, son muy deficientes al lograr una precisión no mayor del 75 %.
Al reflexionar en el muestreo de la distribución se pensó que los pixeles dentro de una vecindad por si mismos son datos muestreados del modelo que los produjo, i.e. ya se tenía el muestreo sin requerir definir siquiera el modelo generador. Esto resultaba muy conveniente porque no se realizaba una suposición sobre la distribución generadora como antes se había hecho y se planteaba como una distribución normal. Además se reduciría considerablemente el tiempo que se hubiera necesitado para realizar todas aquellas simulaciones de Monte-Carlo.
De esta forma ahora las verosimilitudes para el pixel $x \in \mathcal{U}$ con respecto al k-ésimo GMM en lugar de calcularse como:

$$v_k(x) = \sum_{i=1}^{M} \alpha_i p_i(x|\theta_i)$$

se calcularían, suponiendo que cada pixel dentro de la ventana es i.i.d., de la siguiente manera

$$\tilde{v}_k(x) = \prod_{x \in \text{Ventana}} \sum_{i=1}^{M} \alpha_i p_i(x|\theta_i)$$

[3]Se puede observar que la distancia Jensen-Shannon sería una versión simétrica de la distancia Kullback-Leibler cuando $\alpha = \beta = 0{,}5$

Lo único que resta es finalmente normalizar las verosimilitudes teniendo en cuenta el umbral aplicado en la sección anterior. El tamaño de la ventana dependerá del tipo de texturas que cuenta la imagen por lo que resulta dificil ubicar un tamaño adecuado. Finalmente después de las modificaciones realizadas al algoritmo original, para la modificación propuesta se fijan $\omega = 3$ y $h = 1e^{-6}$ y los pasos de la segmentación son:

1. Leer o crear el trimapa de la imagen a segmentar.
2. Calcular los GMMs de las k regiones a segmentar y modificarlos con ω.
3. Elegir un tamaño de ventana adecuado dependiendo de las texturas presentes en la imagen para el cálculo de la verosimilitud.
4. Evaluar las verosimilitudes de cada pixel $x \in \mathcal{U}$ tomando en cuenta el valor del umbral h.
5. Iniciar las α's con las verosimiltudes.
6. Elegir los valores de los parámetros μ, λ y γ.
7. Iterar (4.2) cuidando permanecer en la región factible hasta lograr la convergencia o alcanzar el número máximo de iteraciones.
8. Seleccionar al pixel x como parte de la región k si $\alpha_k(x)$ es el máximo de $\alpha(x)$.

4.4.1. Selección del tamaño de ventana

La selección del tamaño de la ventana es sumamente importante debido a que las verosimilitudes calculadas varían de manera significativa de acuerdo a la opción elegida. El tamaño de la ventana debe elegirse de acuerdo a que tan significativa es la textura dentro de una imagen. La elección también dependerá del tamaño de los elementos de la textura presente, si estos son pequeños el tamaño de la ventana deberá ser pequeño (3x3).
En (fig 4.9) se observa una imagen sintética con ruido Gaussiano y su correspondiente trimapa. Definitivamente el color en esta imagen no resulta tan significativo como las distribuciones de tonos de grises que se presentan en ambas regiones de la imagen. Como puede observarse en (fig 4.9), a partir del trimapa propuesto, las verosimilitudes con un tamaño de ventana de 1x1 no proporcionan mucha información sobre las regiones encontradas en la imagen. Sin embargo al aumentar el tamaño de la ventana las verosimilitudes logran caracterizar cada región por completo de manera bastante precisa, de tal forma que que no es necesario realizar un postproceso, como QMPF, para refinar la segmentación.
La figura (4.9) es un buen ejemplo donde la elección del tamaño de la ventana permite reconocer la textura y mejora de forma indiscutible las verosimilitudes cuando el tamaño de ventana es de 1x1. En esta imagen predomina el color café, por lo que el color no es significativo, sin embargo la textura es diferente de la mariposa a la del lugar donde esta reposa. Con un tamaño de ventana de 1x1 las verosimilitudes de las sombras proyectadas por la mariposa y una sombra que aparece en la parte superior

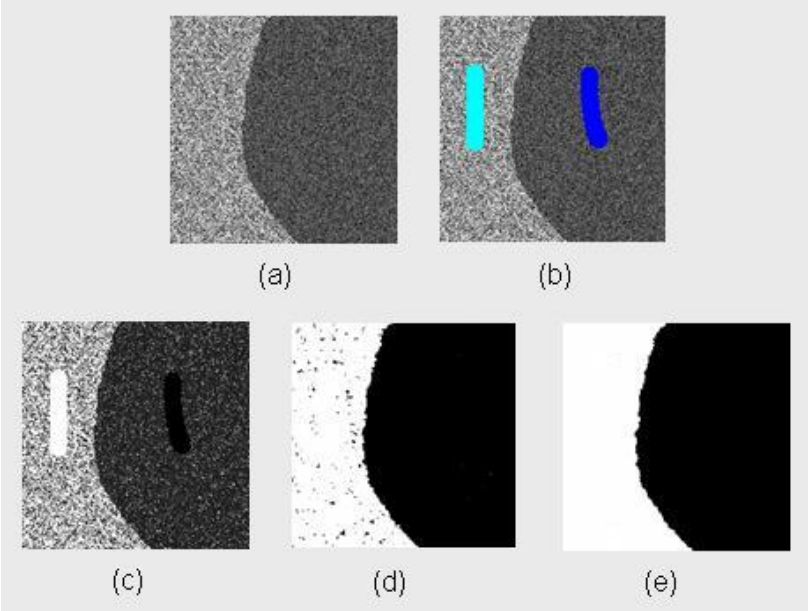

Figura 4.9: (a) Imagen sintética con ruido Gaussiano (b) Trimapa (c) Verosimilitudes obtenidas con un tamaño de ventana de 1x1 (d) Verosimilitudes obtenidas con un tamaño de ventana de 3x3 (e) Verosimilitudes obtenidas con un tamaño de ventana de 5x5

de la imagen, las considera como parte de la mariposa. Con un tamaño de 3x3 se logra identificar casi correctamente a la mariposa evitando clasificar la sombra y mejora además las verosimilitudes de aquellos pixeles que pertenecen a la mariposa.

4.4.2. Alternativa al tamaño de ventana

La idea principal sobre tratar de reconocer la textura en una imagen es, como ya se mencionó anteriormente, que los pixeles dentro de una vecindad son variables aleatorias i.i.d. por lo que la verosimilitud de un pixel se puede calcular como:

$$\tilde{v}_k(x) = \frac{\prod_{x\in\text{Ventana}} \sum_{i=1}^{M} \alpha_{(k,i)} p_{(k,i)}(x|\theta_{(k,i)})}{\sum_{k=1}^{K} \left(\prod_{x\in\text{Ventana}} \sum_{i=1}^{M} \alpha_{(k,i)} p_{(k,i)}(x|\theta_{(k,i)}) \right)} \tag{4.18}$$

Una medida que nos indica la distancia de que un pixel haya sido generado por un modelo puede ser la siguiente

$$d_k(x) := -\log(v_k(x)).$$

Con esta definición de medida podemos despejar a la verosimilitud quedando

$$v_k(x) = \exp(-d_k(x)).$$

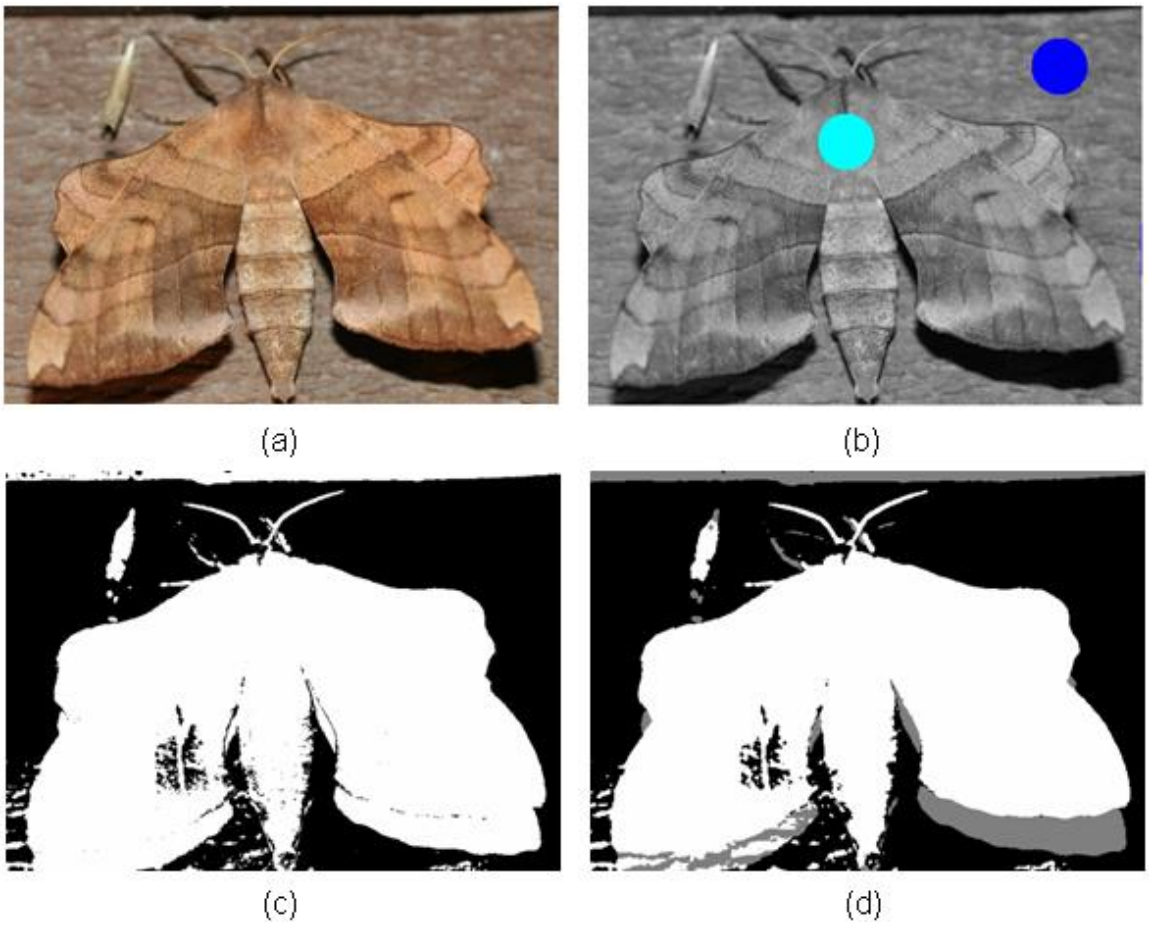

Figura 4.10: (a) Imagen real (b) Trimapa (c) Verosimilitudes obtenidas con un tamaño de ventana de 1x1 (d) Verosimilitudes obtenidas con un tamaño de ventana de 3x3

Si tomamos a la verosimilitud calculada como (4.18) entonces la ecuación anterior toma la siguiente forma

$$\tilde{v}_k(x) = \frac{\prod_{x\in\text{Ventana}} \exp(-d_k(x))}{\sum_{k=1}^{K}\left(\prod_{x\in\text{Ventana}} \exp(-d_k(x))\right)} = \frac{\exp\left(-\sum_{x\in\text{Ventana}} d_k(x)\right)}{\sum_{k=1}^{K}\left(\exp\left(-\sum_{x\in\text{Ventana}} d_k(x)\right)\right)} \quad (4.19)$$

En la fig (4.11) se observa que las verosimilitudes calculadas para diferentes tamaños

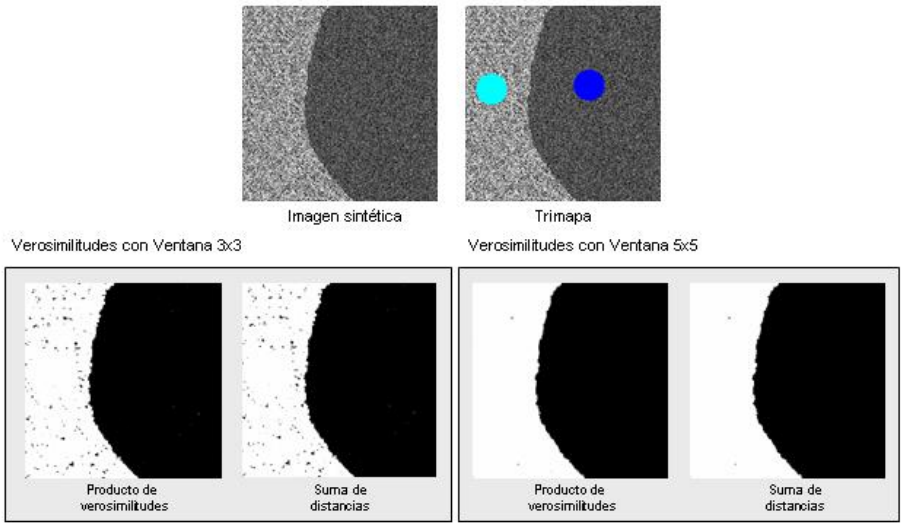

Figura 4.11: Presenta una imagen con su trimapa y se demuestra que la alternativa propuesta genera el mismo resultado en el cálculo de las verosimilitudes

de ventana son las mismas que se obtienen con la manipulación de las distancias.
La sumatoria de distancias en (4.19), se puede observar como un filtro lineal cuya máscara de convolución es la siguiente

1	1	1
1	1	1
1	1	1

Para generalizar esa suma de distancias se define la siguiente suma pesada

$$D(x) = \sum_{y\in\mathcal{N}_x} w_{xy} d(y)$$

donde w_{xy} es el peso que se le dá a cada uno de los pixeles vecinos que conforman la vecindad $\mathcal{N}_x$, la cual a su vez representa una ventana de tamaño prefijado. De esta

forma el producto de las verosimilitudes dentro de un vecindad esta dado por

$$\begin{aligned}\exp(-D(x)) &= \exp\left(-\sum_{y\in\mathcal{N}_x} w_{xy}d(y)\right)\\ &= \prod_{y\in\mathcal{N}_x}\exp\left(-w_{xy}d(y)\right)\\ &= \prod_{y\in\mathcal{N}_x}\left(\exp\left[-d(y)\right]\right)^{w_{xy}}\\ &= \prod_{y\in\mathcal{N}_x} v(x)^{w_{xy}} \qquad (4.20)\end{aligned}$$

En (4.20) puede observarse la relación que existe entre cualquier tipo de filtro lineal, que puede representarse a través de una máscara de convolución, con el producto de las verosimilitudes. Estos pesos en el producto de las verosimilitudes se pueden interpretar como la preferencia que se asigna a cada pixel dentro de la ventana prefijada.

La fig (4.12) resume en un diagrama de flujo los pasos que deben de seguirse para

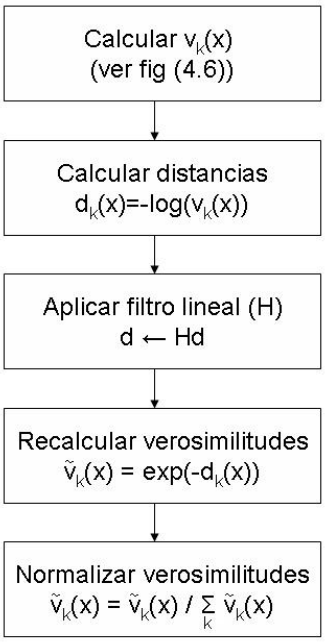

Figura 4.12: Diagrama de flujo sobre el filtrado de las distancias

obtener las verosimilitudes robustas ante la textura como alternativa al tamaño de ventana.

Un filtro lineal comunmente usado es el binomial cuya máscara de convolución es

1	2	1
2	4	2
1	2	1

La fig(4.13) muestra las verosimilitudes calculadas utilizando la máscara de convolución binomial en lugar de la suma de distancias de (4.19). En este cálculo como puede apreciarse en la máscara de convolución se le da mayor importancia al pixel central, posteriormente a los vecinos de primer orden y finalmente al resto de los que forman la vecindad.

Otro filtro lineal es la difusión Homogénea que puede seguir un esquema Gauss-Seidel

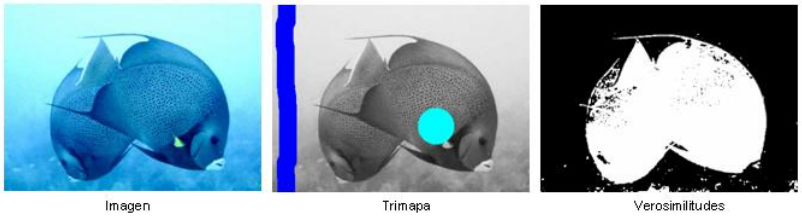

Figura 4.13: En esta imagen se muestra la imagen, su trimapa y las verosimilitudes calculadas a partir del filtrado lineal binomial de las distancias.

cuyos parámetros son η que es el parámetro de difusión y el número de iteraciones y está dado por

$$f(x) = \frac{g(x) + \eta \cdot \sum_{y \in \mathcal{N}_x} f(y)}{1 + \eta \cdot \#\mathcal{N}_x};$$

donde $g(x)$ es la imagen original y $f(x)$ la imagen filtrada.

La máscara de convolución de este filtro lineal puede calcularse aplicando la difusión Homogénea sobre la imagen de un pulso (ver fig (4.14)).

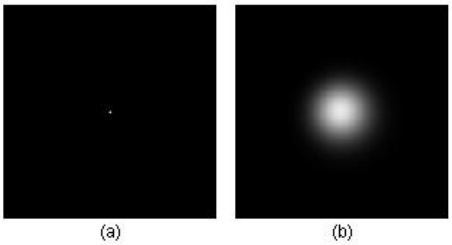

Figura 4.14: (a)Impulso (b)Máscara de convolución de una difusión Homogénea con $\eta = 50$ y iteraciones $= 120$

Capítulo 5

Resultados y discusión

A continuación se presentará una comparación cuantitativa del algoritmo propuesto con respecto al estado del arte sobre un grupo de imágenes de un benchmark. También se demostrará la capacidad de tratar con texturas, aún cuando el algoritmo no fue creado con ese objetivo. Finalmente se darán algunas conclusiones y se hablará de un posible trabajo futuro.

5.1. Comparación cuantitativa con respecto al estado del arte

En el capítulo anterior se dejaron abiertas dos posiblidades del algoritmo propuesto, el número de características extras que podrían agregarse para realizar el clustering con el EM MDL y el tamaño de la ventana para la robustez ante la textura. Estas no se definieron anteriormente porque como se mencionó dependen del tipo de imágenes que quieran segmentarse y por lo mismo se eligirán de acuerdo a las imágenes del *benchmark* con el que se trabajará. A continuación se presentan las elecciones sobre éstas características pendientes y una vez elegidas estas, se compara el método propuesto con el estado del arte.

5.1.1. Elecciones finales sobre características del algoritmo

Para realizar una comparación cuantitativa del algoritmo propuesto con respecto al estado del arte se trabajó con el *benchmark* de Lasso que es una base de datos de imágenes que se encuentra disponible en [1] (ver fig (5.1)). Este *benchmark* contiene un conjunto de imágenes reales con su correspondiente trimapa y la respectiva segmentación realizada por un experto. Un trimapa de Lasso es una imagen donde cada pixel tiene asignada una y sólo una de las siguientes etiquetas:

- ($\mathcal{M}$): Región sin interés.
- ($\mathcal{B}$): Fondo de la imagen.
- ($\mathcal{U}$): Región desconocida donde cada $x \in \mathcal{U}$ debe segmentarse en $x \in \{\mathcal{B} \vee \mathcal{F}\}$.
- ($\mathcal{F}$): Región de interés.

Figura 5.1: Ejemplo de una imagen del *benchmark* de Lasso. (a) Imagen (b) Trimapa (c) Segmentación manual

Antes de realizar la comparación cuantitativa deben de buscarse el número adecuado de características y el tipo de características que deben incluirse para el clustering, al igual que el tamaño de ventana apropiado. Dado que la textura es considerada con el tamaño de la ventana para calcular la verosimilitud, entonces se optó por agregar como rasgos extras de la imagen los bordes.
Los bordes se calcularon de dos formas diferentes que fueron:

1. *Norma de las Derivadas Gaussianas de la imagen.* Primero se convirtió la imagen en blanco y negro, se normalizó y se calcularon las derivadas Gaussianas con $\sigma = 1$. Finalmente se calculó la norma Euclidiana para obtener los bordes (ver fig(5.2)).

2. *Norma de los términos de regularización cálculados como (3.19).* Para obtener los bordes se transformó la imagen al espacio de color CIE-Lab con la librería de Ruzon [53]. Posteriormente cada dimensión se suavizó aplicando tan sólo tres iteraciones de un filtro de membrana homogéneo para después calcular los términos de regularización siguiendo un esquema de diferencias finitas como en (3.19) con $\gamma = 1{,}3e^{-6}$. Por último se calculó la norma Euclidiana para obtener los bordes (ver fig(5.3)).

Lamentablemente el ingreso de los bordes como un rasgo extra para el clustering no mejoró los resultados, debido a que en comparación con la distribución del color la información de los bordes al parecer no resulta significativa al menos para el *benchmark* utilizado.
Lo que restaba era definir el tamaño de ventana para calcular las verosimilitudes (ver Sección 4.4). Para esto se probaron con tres tamaños diferentes de ventanas de 1x1, 3x3 y 5x5 con el *benchmark* de Lasso (ver fig (5.4)). Como puede apreciarse en la fig (5.4) mientras mayor sea el tamaño de la ventana más definida será la segmentación previa, pero eso no implica que los resultados sean siempre los deseados porque se pueden definir aún más también regiones erróneas. Al haber comparado los resultados a partir de las distintas verosimilitudes obtenidas de cada tamaño de ventana, se optó por elegir el tamaño de ventana de 3x3.

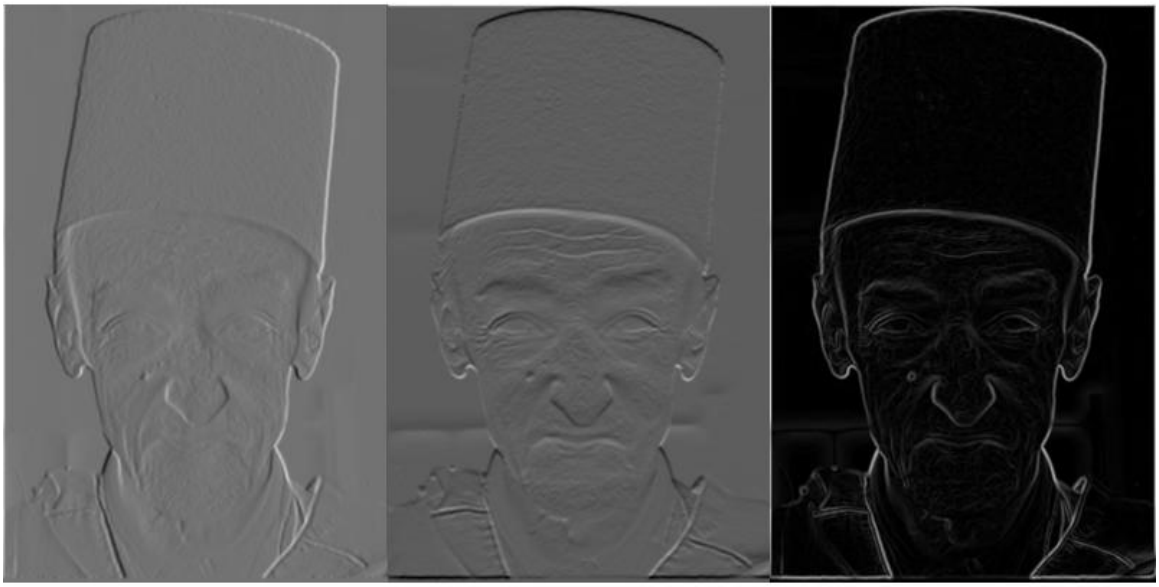

Figura 5.2: Bordes obtenidos con la norma de las derivadas Gaussians. (a) Dx Gaussiana (b) Dy Gaussiana (c) Norma de las derivadas Gaussianas

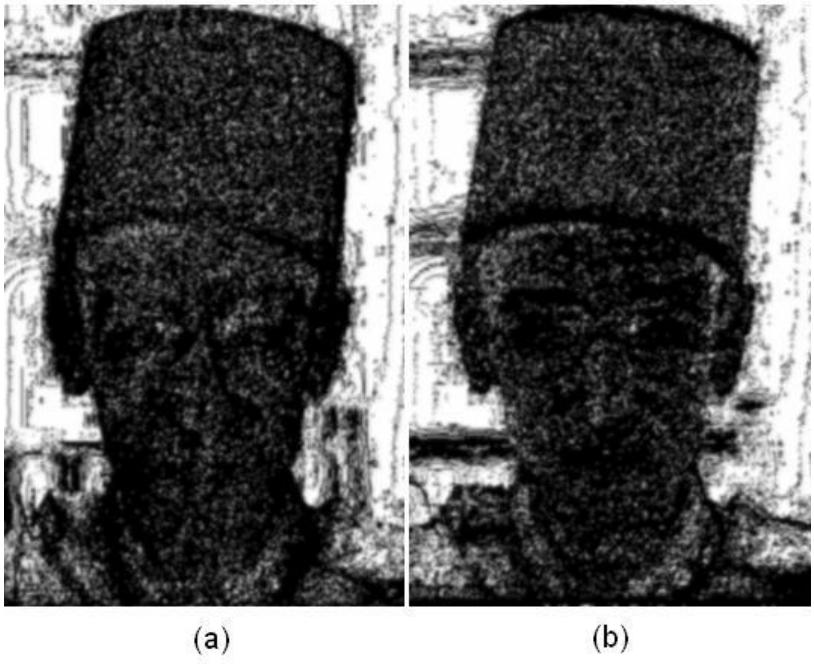

Figura 5.3: Bordes a partir de los términos de regularización del algoritmo QMPF. (a) Bordes en x $l_{x,x+1}$ (b) Bordes en y $l_{y,y+1}$

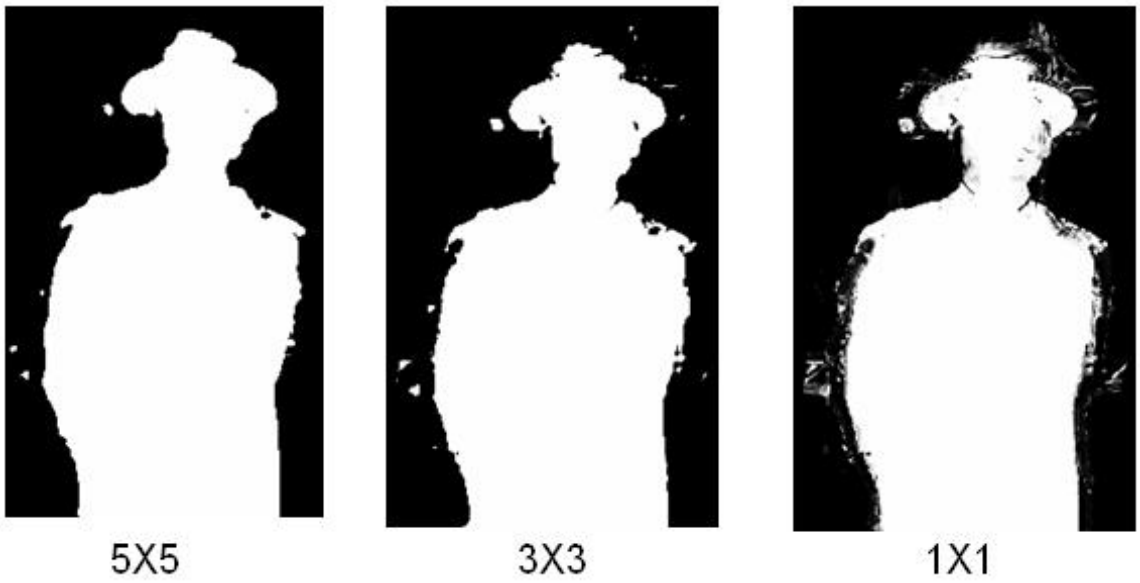

Figura 5.4: Verosimilitudes de fig (5.1) obtenidas a partir de distintos tamaños de ventana

La fig (5.5) es un ejemplo de la segmentación lograda de una imagen del *benchmark* de Lasso. En esta se presenta la imagen, su trimapa, las verosimilitudes calculadas con un tamaño de ventana de 3x3, los bordes que se utilizan como término de regularización individuales de la función de costo del QMPF y finalmente la segmentación final.

5.1.2. Tablas comparativas

Para llevar a cabo la comparación cuantitativa de los resultados obtenidos por el estado del arte y este algoritmo primero deben ajustarse los parámetros que mejor segmentan las 50 imágenes del *benchmark* de Lasso. Para lograrlo se utilizó el algoritmo de descenso Nelder Mead [43] por tratarse de una función no diferenciable que debe optimizarse. Esta función cuenta con los parámetros (γ (para los términos de regularización basados en bordes), λ (para promover regiones suaves) y μ (para controlar la entropía)) y su imagen es el error promedio de todas las imágenes. Los parámetros aprendidos utilizando el QMPF sin control de entropía son ($\lambda = 16378463{,}37148, \gamma = 23{,}160950$) y utilizando el QMPF con control de entropía son ($\lambda = 15357902{,}332238, \gamma = 23{,}171794, \mu = -103{,}918976$).
La comparación del método propuesto se realizó con el estado del arte para la segmentación binaria de imágenes: máximo flujo (*Graph Cut*), GMMF y Caminata aleatoria. También se comparó con respecto al desempeño del QMPF anterior con control y sin control de entropía. Las tablas (5.1) y (5.1) muestran los errores porcentuales que se tuvieron en cada imagen del *benchmark* de Lasso con respecto a cada uno de los

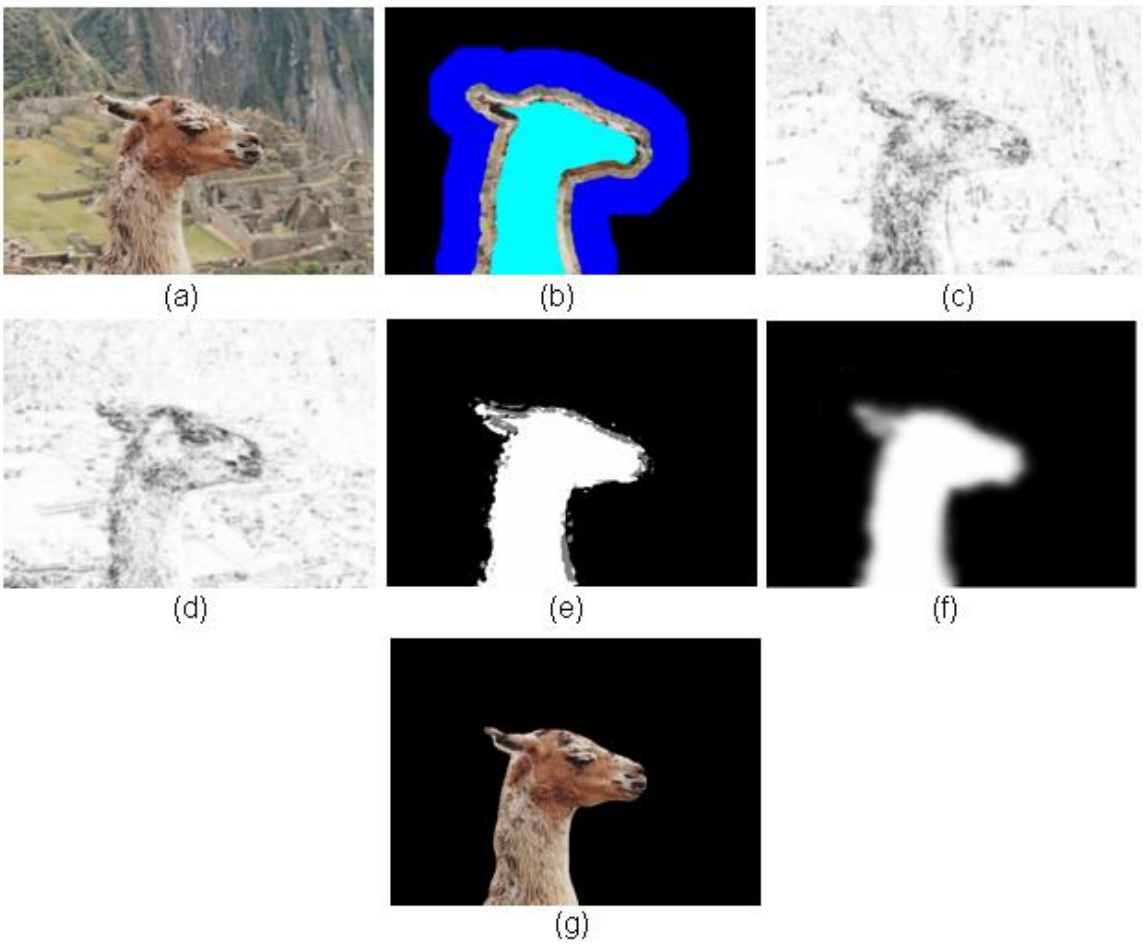

Figura 5.5: (a) Imagen del *benchmark* de Lasso (b) Trimapa de Lasso (c) Bordes en x (d) Bordes en y (e) Verosimilitud obtenida con tamaño de ventana de 3x3 (f) Resultado del QMPF (g) Segmentación final

métodos[1]. Cuando se calcularon los errores del QMPF propuesto sin control de en-

Imagen	**GraphCut**	**GMMF**	**RW**	**QMPF (original)**	**QMPF (original) + EC**	**QMPF (propuesto)**	**QMPF (propuesto) + EC**
21077	7.82 %	3.84 %	3.77 %	3.84 %	4.01 %	3.53 %	3.55 %
24077	16.68 %	10.44 %	10.48 %	11.00 %	4.21 %	5.18 %	5.17 %
37073	8.84 %	5.40 %	5.43 %	5.06 %	1.44 %	1.17 %	1.19 %
65019	6.49 %	2.32 %	2.20 %	3.68 %	0.27 %	0.56 %	0.70 %
69020	7.99 %	6.29 %	6.28 %	5.80 %	2.95 %	6.92 %	6.92 %
86016	7.49 %	3.39 %	3.37 %	3.01 %	1.99 %	1.98 %	1.98 %
106024	10.05 %	8.89 %	8.93 %	8.20 %	7.55 %	9.29 %	9.35 %
124080	3.57 %	5.14 %	5.22 %	3.40 %	3.43 %	5.98 %	5.99 %
153077	13.04 %	3.99 %	3.97 %	6.32 %	1.65 %	1.40 %	1.33 %
153093	4.75 %	4.30 %	4.35 %	3.70 %	4.08 %	6.06 %	6.07 %
181079	13.42 %	13.16 %	13.14 %	12.19 %	7.41 %	8.37 %	8.32 %
189080	11.66 %	10.79 %	10.77 %	10.44 %	6.22 %	8.00 %	7.93 %
208001	3.53 %	3.21 %	3.23 %	2.27 %	1.50 %	1.39 %	1.39 %
209070	5.98 %	3.86 %	3.85 %	3.82 %	2.25 %	1.90 %	1.91 %
227092	3.99 %	2.40 %	2.37 %	3.90 %	3.46 %	4.58 %	4.53 %
271008	4.68 %	3.43 %	3.48 %	3.47 %	2.33 %	3.38 %	3.39 %
304074	16.69 %	6.10 %	6.12 %	8.29 %	10.90 %	9.83 %	9.83 %
326038	11.30 %	14.73 %	14.87 %	9.00 %	7.53 %	10.08 %	10.08 %
376043	15.95 %	11.06 %	11.05 %	11.47 %	6.14 %	4.97 %	4.97 %
388016	1.64 %	3.45 %	3.51 %	2.39 %	1.50 %	3.79 %	3.81 %
banana1	12.21 %	10.45 %	10.44 %	10.31 %	3.91 %	1.67 %	1.63 %
banana2	2.81 %	5.08 %	5.06 %	3.41 %	1.49 %	1.39 %	1.39 %
banana3	5.69 %	8.04 %	8.10 %	5.53 %	1.91 %	0.43 %	0.43 %
book	8.24 %	9.99 %	10.01 %	7.99 %	3.52 %	1.89 %	1.71 %
bool	4.01 %	2.00 %	2.00 %	2.17 %	1.74 %	3.81 %	3.79 %
bush	13.02 %	9.12 %	9.04 %	11.28 %	7.86 %	8.09 %	8.09 %
cercamic	4.92 %	6.18 %	6.18 %	4.97 %	1.73 %	3.68 %	3.54 %

Cuadro 5.1: Parte 1 de la comparación de varios métodos de segmentación en el *benchmark* de Lasso

tropía, se crearon grandes expectativas sobre la mejora del método dado que vencía en casi un punto porcentual al QMPF anterior sin control de entropía. Lamentablemente cuando se incluyó el control de entropía los resultados no mejoraron y fueron vencidos por el QMPF + EC.

[1]Todos los errores que se presentan en las tablas, excepto los de los métodos propuestos, se tomaron de [50]

Imagen	**GraphCut**	**GMMF**	**RW**	**QMPF (original)**	**QMPF (original) + EC**	**QMPF (propuesto)**	**QMPF (propuesto) + EC**
cross	3.30 %	2.35 %	2.35 %	2.36 %	1.75 %	16.47 %	16.36 %
doll	2.53 %	2.92 %	2.91 %	2.28 %	1.03 %	0.82 %	0.82 %
elephant	8.51 %	6.19 %	6.13 %	5.83 %	2.05 %	2.39 %	2.24 %
flower	1.23 %	0.71 %	0.71 %	0.58 %	0.61 %	0.45 %	0.45 %
fullmoon	0.95 %	0.88 %	0.88 %	0.80 %	0.27 %	0.26 %	0.09 %
grave	1.84 %	1.86 %	1.89 %	1.44 %	1.27 %	4.03 %	4.03 %
llama	18.30 %	14.65 %	14.69 %	13.86 %	4.32 %	3.98 %	3.98 %
memorial	9.93 %	4.75 %	4.75 %	5.72 %	1.49 %	1.69 %	1.68 %
music	3.43 %	2.21 %	2.19 %	2.70 %	2.26 %	2.13 %	2.13 %
person1	3.06 %	4.55 %	4.69 %	2.33 %	1.16 %	2.21 %	2.56 %
person2	1.35 %	4.48 %	4.54 %	1.24 %	0.71 %	0.59 %	0.59 %
person3	6.62 %	4.81 %	4.81 %	3.63 %	0.87 %	4.14 %	4.14 %
person4	8.46 %	6.27 %	6.23 %	5.88 %	3.27 %	4.43 %	4.42 %
person5	7.51 %	4.62 %	4.62 %	4.77 %	2.48 %	8.55 %	8.58 %
person6	9.87 %	7.91 %	7.88 %	8.37 %	5.19 %	4.92 %	4.91 %
person7	1.41 %	2.21 %	2.26 %	1.31 %	0.96 %	1.38 %	1.39 %
person8	6.15 %	3.92 %	3.88 %	3.92 %	0.93 %	0.95 %	0.95 %
scissors	5.89 %	6.38 %	6.32 %	6.52 %	2.87 %	2.80 %	2.81 %
sheep	2.48 %	3.08 %	3.14 %	1.83 %	4.53 %	7.43 %	7.43 %
stone1	1.08 %	0.72 %	0.72 %	0.79 %	0.73 %	1.73 %	1.60 %
stone2	0.56 %	0.33 %	0.32 %	0.49 %	0.78 %	2.31 %	2.33 %
teddy	2.19 %	2.91 %	2.95 %	2.23 %	1.91 %	2.97 %	3.19 %
tennis	8.69 %	7.55 %	7.61 %	8.34 %	7.31 %	11.71 %	11.73 %
media	6.84 %	5.47 %	5.47 %	5.08 %	3.03 %	4.15 %	4.15 %
mediana	6.07 %	4.59 %	4.66 %	3.87 %	2.15 %	3.45 %	3.46 %
desvest	4.66 %	3.57 %	3.58 %	3.46 %	2.40 %	3.43 %	3.43 %

Cuadro 5.2: Parte 2 de la comparación de varios métodos de segmentación en el *benchmark* de Lasso

Cabe mencionar que aunque los resultados no fueron tan alentadores, finalmente el método propuesto, al igual que los otros métodos, pertenece al grupo de métodos de segmentación asistida cuyo propósito es que el usuario participe en el proceso de segmentación con la elaboración del trimapa. Por lo tanto la inclusión de la robustez ante textura resulta una mejora, probablemente si el *benchmark* contara con imágenes con mayor textura el método propuesto hubiera tenido mejores resultados. La suposición anterior podría justificarse en la fig (5.5) donde la textura juega un papel importante y por lo tanto el error por el método propuesto es menor que el obtenido por el resto de los métodos (ver error porcentual de *llama* en la tabla (5.2)).

5.2. Segmentación de texturas

Aún cuando la inclusión de la textura en el proceso de segmentación no se realizó con el fin de segmentar exclusivamente texturas, se realizaron algunas pruebas sobre mosaicos de texturas. Los resultados fueron muy buenos y pueden apreciarse en la fig (5.6) donde se presentan las verosimilitudes obtenidas para distintos tamaños de ventanas, la segmentación obtenida con las máximas verosimilitudes (ML) y la segmentación lograda por el QMPF sin términos de regularización individuales (i.e. sin la información de los bordes).
Como puede observarse, las verosimilitudes mejoran conforme crece el tamaño de ventana de 1x1 hasta 7x7. Después del tamaño de la ventana de 7x7 los resultados empeoran hasta que no pueden definir ningún pixel como parte de alguna clase. Cabe destacar que la segmentación lograda por la ML para el tamaño de ventana de 7x7 son mejores que las logradas por el QMPF para cualquier otro tamaño de ventana, lo cual demuestra la ventaja que tiene elegir un tamaño de ventana adecuado para lograr percibir la textura.
También debe señalarse el poder que tiene el QMPF para segmentar imágenes, debido a que aún cuando aparentemente las verosimilitudes obtenidas a partir de un tamaño de ventana de 1x1 parecen no proporcionar mayor información, después de aplicar el QMPF se logran definir bastante bien las regiones con una misma textura. Debido a que las segmentaciones nos definían múltiples regiones y no solamente aquellas con texturas diferentes se decidió incluir dos rasgos extras, al momento de calcular las mezclas de Gaussianas, que son la posición (i, j) de cada pixel. En la fig (5.7) puede observarse como la inclusión de la posición al momento del clustering influye positivamente en el cálculo de las verosimilitudes dando resultados muy buenos incluso cuando solamente se está aplicando la ML para la segmentación. Aún cuando estos resultados son mejores que los mostrados en la fig (5.6), lamentablemente las fronteras de las regiones se deforman a causa de la influencia que tiene la posición al momento de la estimación de las distribuciones.
Los resultados al aplicar la difusión homogénea, propuesta en la sección 4.4.2, mejoran los resultados observados en la fig(5.7). La acción de la difusión sobre la distancias evita utilizar la posición (i, j) como rasgos extras lo cual resulta muy benificioso debido a que en muy pocos casos, que dependen del trimapa, conviene hacerlo.

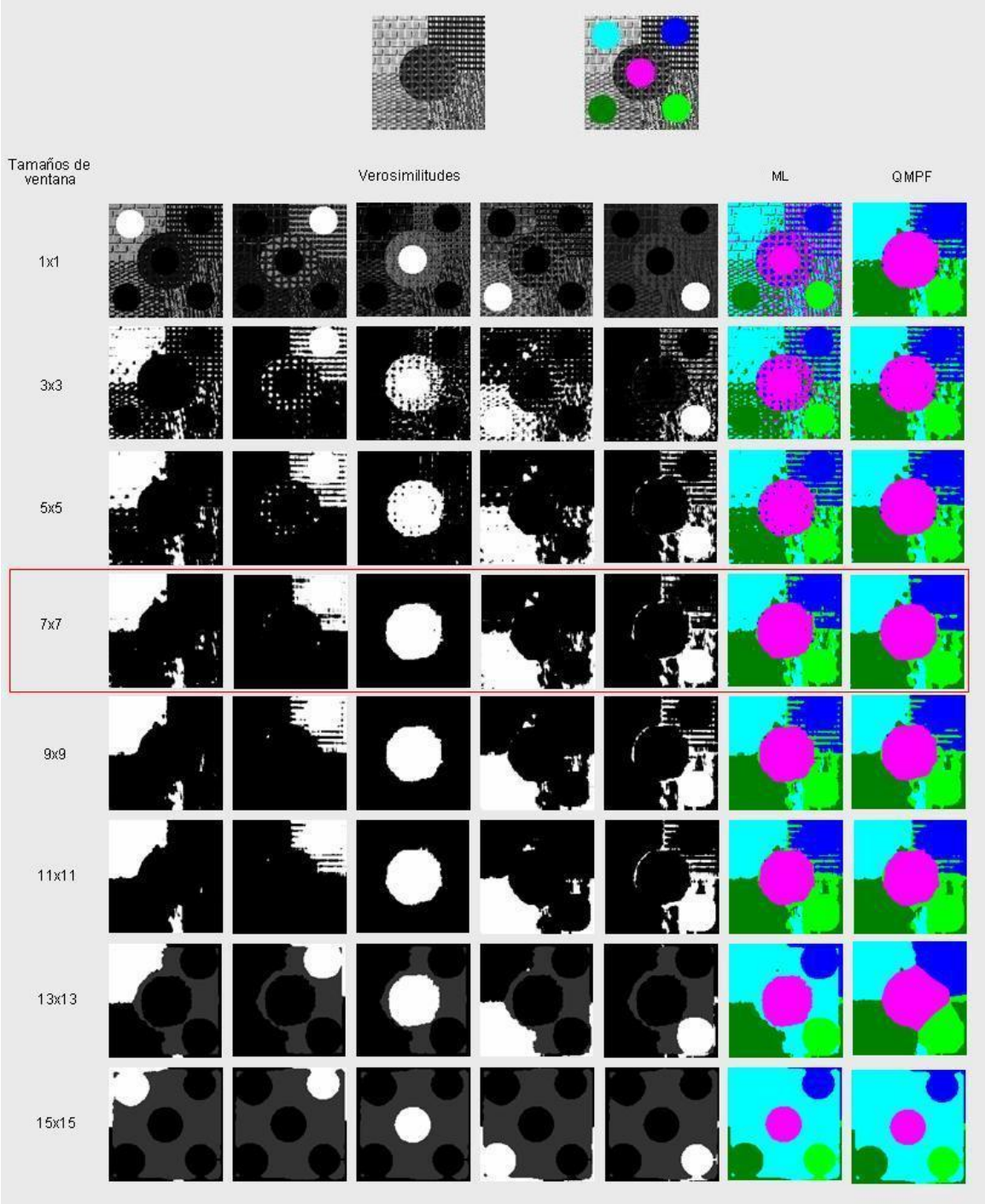

Figura 5.6: Segmentación de texturas para distintos tamaños de ventanas

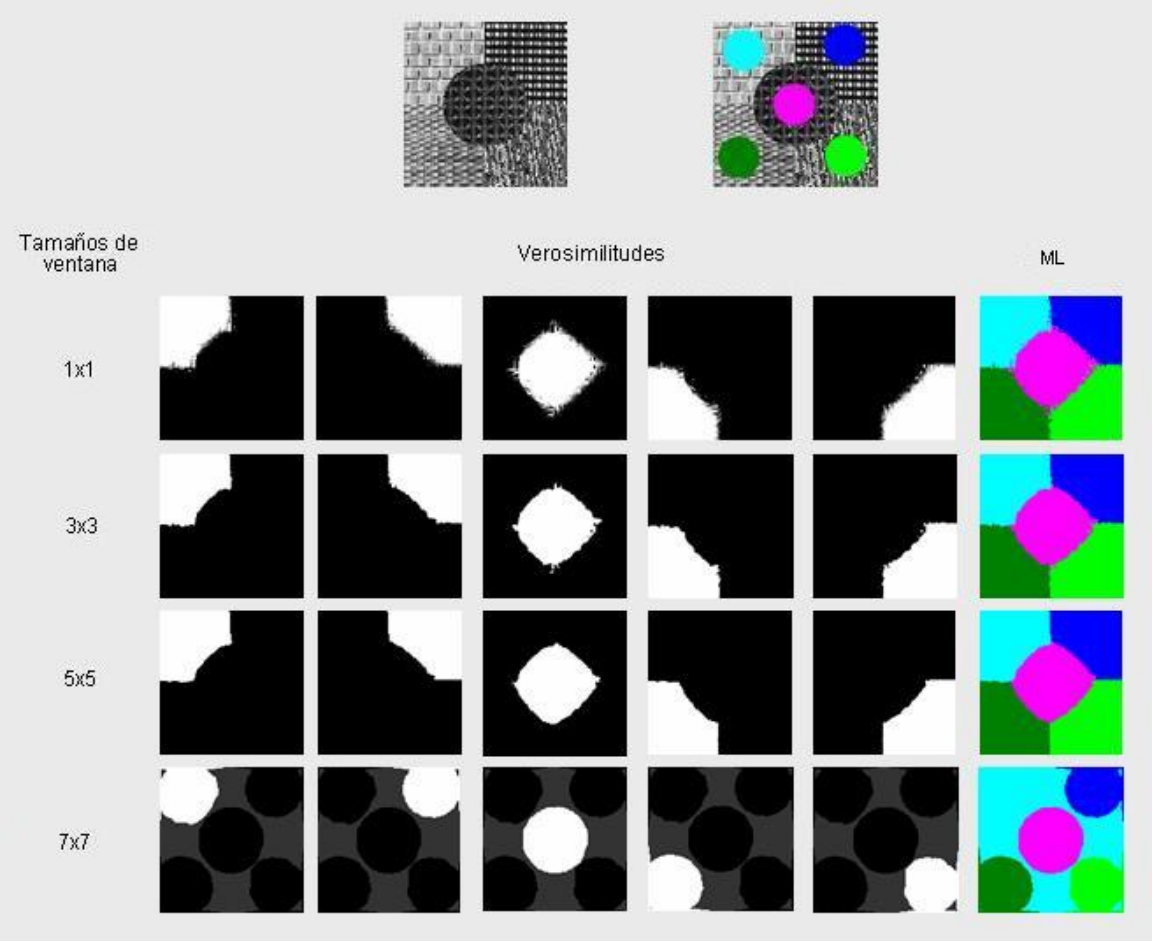

Figura 5.7: Segmentación de texturas para distintos tamaños de ventanas con la posición del pixel como dos rasgos extras

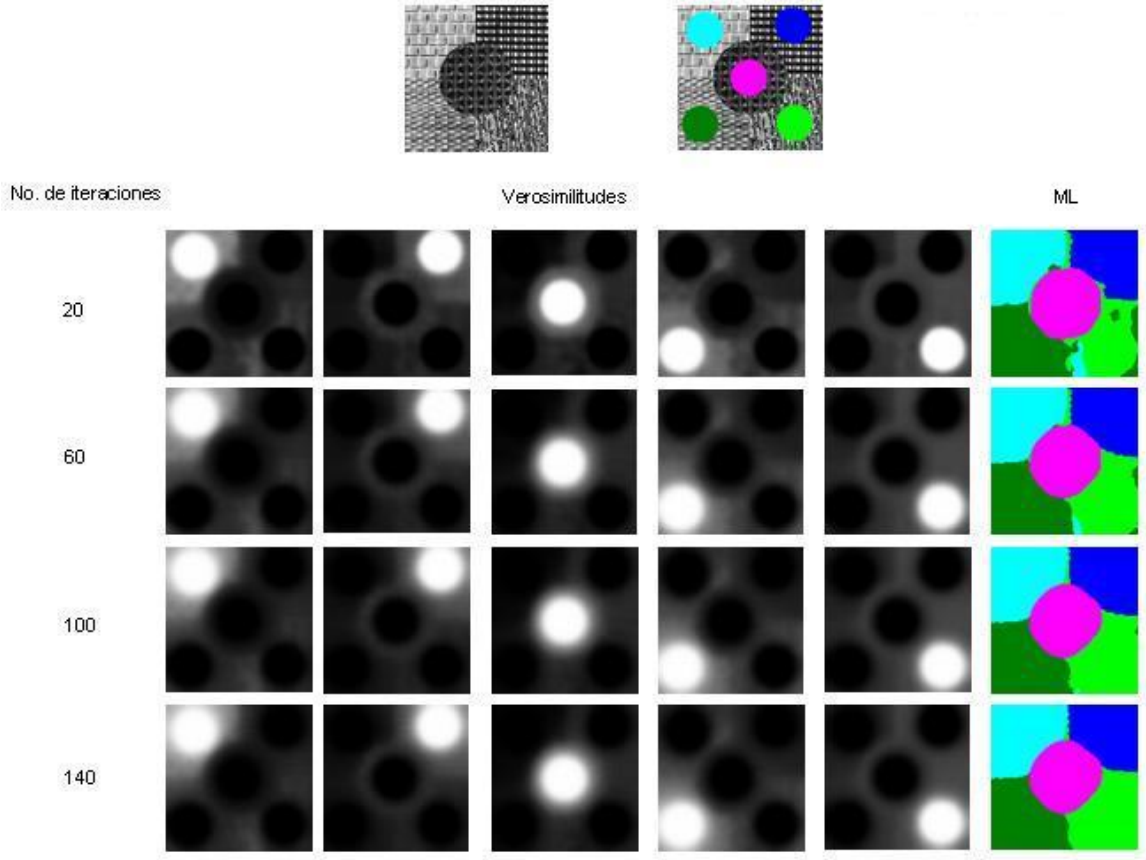

Figura 5.8: Segmentación de texturas aplicando una difusión Homogénea con $\eta = 50$

Capítulo 6

Conclusiones y Trabajo a futuro

6.1. Conclusiones

La principal motivación de este trabajo fue la de extraer mayor información de la imagen, no solamente la proporcionada por los canales de color, para mejorar el proceso de segmentación asistida realizada con QMPF.
El uso de histogramas como herramienta para calcular la distribución de forma no parámetrica de las K regiones marcadas por el multimapa proporcionaba un cálculo adecuado de las verosimilitudes de los pixeles [50]. Sin embargo cuando aumenta la dimensión del problema se vuelve complicada su manipulación y un problema con la memoria. Siguiendo la propuesta de [24] de utilizar una mezcla de Gaussianas en lugar del uso de histogramas, se implementó el algoritmo EM-MDL que nos permitía calcular de forma no parámetrica el número de distribuciones, al igual que los histogramas, de forma rápida y eficiente.
Utilizar la mezcla de gaussianas acabó con el problema de la memoria y facilitó enormemente su manipulación, pero las verosimilitudes calculadas, aunque aparentemente muy buenas, eran casi binarias por lo que el post-proceso no realizaba ninguna mejora sobre la segmentación prematura. Para evitar este problema se modificó la matriz de Covarianza agregándole incertidumbre y se construyó un umbral que cuando las verosimilitudes calculadas son menores a este se asigna como si las verosimilitudes de ese pixel fueran iguales para todas las clases. Con la implementación de lo anterior se lograron obtener verosimilitudes muy buenas para posteriormente aplicar el QMPF (ver fig(4.7) y fig(4.8)).
Debido a que se buscaba hacer la segmentación robusta ante la textura, se ideó una propuesta sencilla y novedosa para poder lograrlo que consiste en considerar cada pixel dentro de una vecindad previamente definida como una variable aleatoria i.i.d.. De tal forma que la verosimilitud de un sólo pixel x se calcularía como el producto de las verosimilitudes individuales de cada pixel que forma parte de la vecindad $mathcal N_x$.
Los mejores resultados obtenidos fueron sobre aquellas texturas cuyos primitivos son pequeños.
Se presentó también una propuesta paralela al uso de un tamaño de ventana para trabajar sobre las distancias y se explican las implicaciones de distintas convolucio-

nes sobre el producto de las verosimilitudes en una tamaño de ventana prefijado.
Los resultados obtenidos a partir de la adición de la textura de forma indirecta en el cálculo de las verosimilitudes resultaron muy buenos entre imágenes con cierta textura (ver fig(4.10)). Para probar la eficiencia de esto se trabajó con la segmentación de exclusivamente texturas, para lo cual no fue ideado, y arrojó resultados que no se hubieran esperado. Cabe señalar que incluso la segmentación lograda por la ML bajo cierto tamaño de ventana logra vencer notoriamente a la segmentación lograda por el QMPF cuando no se toma en cuenta el tamaño de ventana (ver fig(5.6)).
Cabe señalar que la propuesta para el cálculo de las verosimilitudes tiene un impacto no solamente para nuestro método sino para cualquier otro que requiera del cálculo de verosimilitudes.
Las comparaciones cuantitativas con respecto a otros métodos no fueron las deseadas y ocupó el segundo lugar para el *benchmark* de Lasso[1]. Sin el control de la entropía se venció al QMPF-anterior por casi un punto porcentual, cuando se lleva a cabo el control de la entropía los resultados del QMPF-anterior mejoran significativamente mientras que el QMPF-propuesto no lo hace. Sin embargo aquellas imágenes con una textura significativa mejoran su segmentación con respecto al resto de los métodos (ver fig(5.5) y Tabla (5.2)).
Entre las propiedades que se buscaban de la segmentación asistida (ver Capítulo 2) se encuentra la *robustez* la cual no se lograba con el QMPF-anterior y con el QMPF-propuesto se tiene (ver fig(4.9)). Sin embargo no siempre es conveniente trabajar con un tamaño de ventana mayor que 1x1 aún cuando exista textura presente en la imagen (ver fig(6.1)), pero en ese caso se toma en cuenta, otra propiedad de la segmentación asistida, la *interactividad* de la que el usuario hace uso para determinar si es conveniente incluir la textura como parte del proceso de segmentación.

6.2. Trabajo a Futuro

A continuación se presentan algunas propuestas de trabajo a futuro:

- Trabajar con imágenes multibanda satelitales que cuentan de 7 a 20 bandas para aprovechar el uso de la modelación de las distribuciones de las clases a través de una mezcla de Gaussianas.

- Comparar nuestra propuesta para calcular verosimilitudes robustas ante textura con una segmentación que incluya rasgos de textura al momento del clustering. Lo complicado sería elegir entre los numerosos rasgos aquellos que sean útiles para las imágenes prueba.

- Trabajar con filtros no lineales sobre las distancias y ver la equivalencia que se tiene al trabajar con las verosimilitudes.

[1] El *benchmark* de Lasso cuenta con varios trimapas mal trazados debido a que en muchas casos la clase objeto, a pesar de ser muy grande, no representa una selección de pixeles representativa del mismo (ver las imágenes 10624 y bush con sus respectivos trimapas que se encuentran en [1])

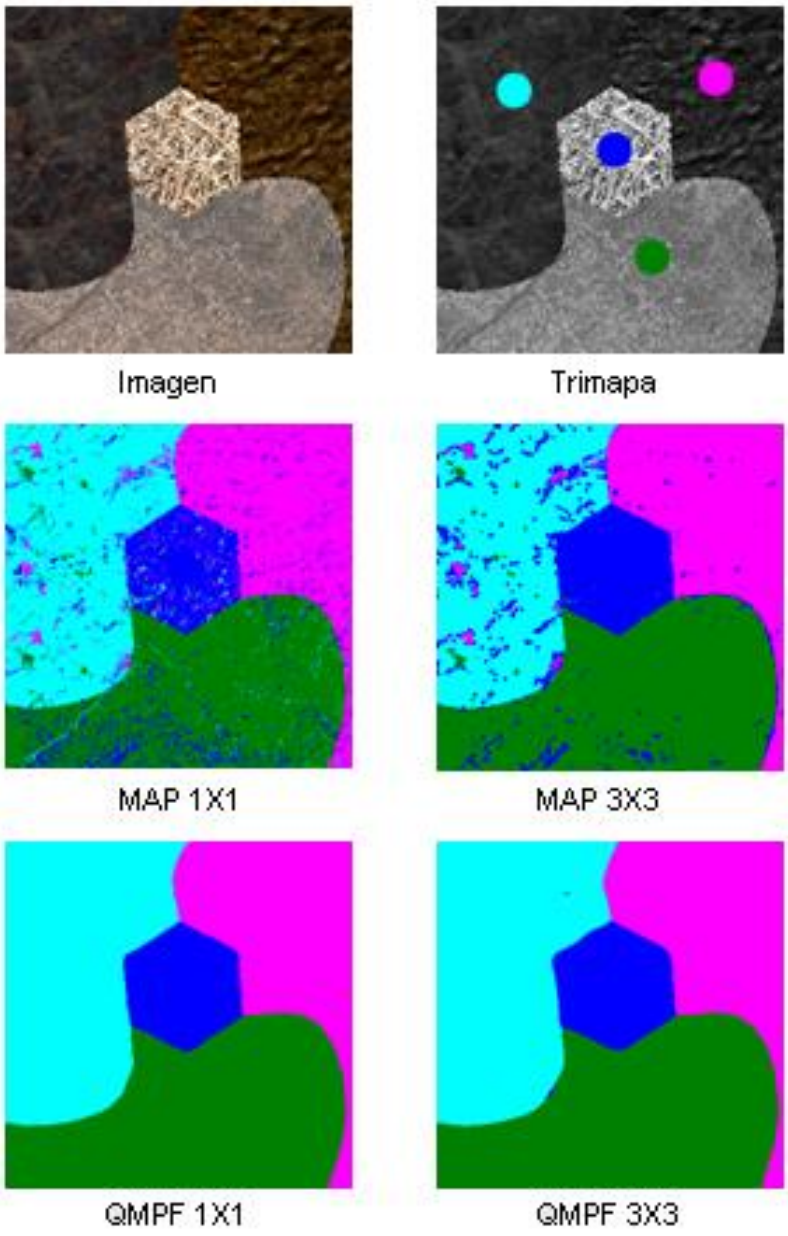

Figura 6.1: En esta imagen se demuestra que a pesar de ser un mosaico de texturas, segmentar únicamente con el color (i.e. tomando un tamaño de ventana de 1x1) arroja mejores resultados. Esto se debe a que la distinción de las regiones por color es visiblemente más sencilla que por texturas.

- Buscar que máscara o máscaras de convolución son las más adecuadas para filtrar las distancias en imágenes de cualquier tipo. Para posteriormente elegir entre estas máscaras de convolución aquella que sea adecuada para el *benchmark* de Lasso y comparar estos nuevos resultados.
- Debido a los buenos resultados que se tienen por la propuesta que se hizo para la robustez ante textura, se plantea la pregunta de que si es necesario un post-proceso, como el QMPF, o el filtrar de forma adecuada las distancias (las menos log-verosimilitud) sea suficiente.

Bibliografía

[1] Lasso's database. *http://research.microsoft.com/vision/cambridge/i3l/segmentation/GraphCut*

[2] M. Brown P. Perez A. Blake, C. Rother and P. Torr. Interactive image segmentation using an adaptive GMMRF model. *Microsoft Research Cambridge UK*, 2004.

[3] T.E. Weymouth A.A. Amini and R.C. Jain. Using dynamic programming for solving variational problems in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(9):855–866, 1990.

[4] C. Roos J.J. Gerbrands A.C.M. Dumay, M.N.A: J Claessens and J.H.C. Reiber. Object delineation in noisy images by a modified policy-iteration method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(9):952–958, 1992.

[5] H. Akaike. A new look at the statistical model identification. *IEEE Trans. Automat. Contr.*, AC(19):716–723, 1974.

[6] D.H. Ballard and C.M.Brown. *Computer Vision*. Prentice Hall, 1982.

[7] D.H. Ballard and J. Sklansky. Tumor detection in radiographs. *Computers and Biomedical Research*, 6(4):299–321, 1973.

[8] W.A. Barrett and B.S. Morse. A relaxation algorithm for segmentation of the endocardial surface from cine ct. In *IEEE Proc. of Computers in Cardiology, pp. 95-98, Jerusalem*, 1989.

[9] R. Bellman and S. Dreyfus. *Applied Dynamic Programming*. Princeton University Press, 1962.

[10] Jeff A. Bilmes. A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. *International Computer Science Institute*, 1998.

[11] Sean Borman. The expectation maximization algorithm - a short tutorial. *http://www.seanborman.com/publications/EM$_a$lgorithm.pdf*, 2006.

[12] Yuri Y. Boykov and Marie-Pierre Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In *Proc. of Internation Conference on Computer Vision*, pages 105–112, 2001.

[13] J.D. Cappelletti and A. Rosenfeld. Three-dimensional boundary following. *Computer Vision, Graphics, and Image Processing*, 48(1):80–92, 1989.

[14] Y.P. Chien and K.S. Fu. A decision function method for boundary detection. *Computer Graphics and Image Processing*, 3(2):125–140, 1974.

[15] W.S. Ching. A novel change detection algorithm using adaptive threshold. *Image and Vision Computing*, 12(7):459–463, 1994.

[16] Michael H. Coen. A similarity metric for spatial probability distributions. *Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology.*

[17] L.D. Cohen. On active contours models and balloons. *CVGIP: Image Understanding*, 53(2):211–218, 1991.

[18] C. Graffigne D. Geman, S. Geman and P. Dong. Boundary detection by constrained optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):609–628, 1990.

[19] J.G. Daugman. Uncertainty relation for resolution in space, spatial-frequency, and orientation optimized by two-dimensional visual cortical filters. *Journal of Optical Society of America*, (2):1160–1169, 1985.

[20] Olivier Duchenne and Jean-Yves Audibert. Fast interactive segmentation using color and textural information. *Research Report CERTIS*, 2006.

[21] J.H. Elder and R.M. Goldberg. Image editing in the contour domain. In *IEEE Proc. of Computer Vision and Pattern Recognition*, pages 374–381, 1998.

[22] Charles A. Bouman et. al. Cluster: An unsupervised algorithm for modeling gaussian mixtures. *School of Electrical Engineering, Purdue University*, 2000.

[23] D. Daneels et al. Interactive outlining: An improved approach using active contours. In *SPIE Proc. of Storage and Retrieval for Image and Video Databases*, pages 226–233, 1993.

[24] S. Belongie et al. Color- and texture- based image segmentation using em and its application to content based image retrieval. In *ICCV*, pages 675–682, 1998.

[25] S. Yu G. Guo and S. Ma. Unsupervised segmentation of color images. In *IEEE Proc. of the Fifth International Conference on Image Processing*, 1998.

[26] T. Gevers and V.K. Kajcovski. Image segmentation by directed region subdivision. In *IEEE Proc. of the 12 International Conference on Pattern Recognition*, 1994.

[27] R.M. Haralick and L.G. Shapiro. *Computer and Robot Vision.* Addison-Wesley, 1992.

[28] B.K.P. Horn. *Robot Vision.* MIT Press, 1986.

[29] Adobe Systems Incorporated. *Adobe Photoshop Version 5.5 Users Guide*, chapter 7: Selecting. 1999.

[30] J.J. Gerbrands I.T. Young and L.J. Van Vliet. *Fundamentals of Image Processing.* Delft University of Technology Press, 1998.

[31] Shiri Gordon Jacob Goldberger and Hayit Greenspan. An efficient image similarity measure based on approximations of KL-divergence between two Gaussian mixtures. In *Proceedings of the Ninth International Conference on Computer Vision*, pages 487–493, 2003.

[32] A.K. Jain and F. Farrokhnia. Unsupervised texture segmentation using gabor filters. *Pattern Recognition*, (24):1167–1186, 1991.

[33] Mojsilovic A. Junqing Chen, Pappas T.N. and Rogowitz B. Adaptive image segmentation based on color and texture. In *ICIP*, pages 770–780, 2002.

[34] K. I. Laws. Textured image segmentation. *Ph.D. Thesis, University of Southern California*, 1980.

[35] Stan Z. Li. *Markov Random Field Modeling in Image Analysis*, pages 8–40. Springer-Verlag, 2001.

[36] Tuceryan M. Moment based texture segmentation. *Pattern Recognition Letters*, 15:659–668, 1994.

[37] V. Hlavac M. Sonka and R. Boyle. *Image Processing, Analysis, and Machine Vision.* PWS Publishing, 1999.

[38] A. Martelli. An application of heuristic search methods to edge and contour detection. *Communications of the ACM*, 19(2):73–83, 1976.

[39] A. Martelli and U. Montanari. Non-serial dynamic programming: On the optimal strategy of variable elimination for the rectangular lattice. *Journal of Mathematical Analysis and Applictions*, 40(1):226–242, 1972.

[40] A. Moghaddamzadeh and N. Bourbakis. A fuzzy region growing approach for segmentation of color images. *Pattern Recognition*, 30(6):867–881, 1997.

[41] U. Montanari. On the optimal detection of curves in noisy pictures. *Communication of the ACM*, 14(5):335–345, 1971.

[42] E.N. Mortensen. Vision-assisted image editing. *Computer Graphics*, 33(4):55–57, 1999.

[43] J.Ñocedal and S. Wright. *Numerical Optimization.* Springer Series in Operational Research, Springer-Verlag, 1999.

[44] N. Otsu. A threshold selection method from grey-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–69, 1979.

[45] D. Medici P. Campadelli and R. Schettini. Color image segmentation using hopfield networks. *Image and Vision Computing*, 5(3):161–166, 1997.

[46] T. Pavlidis and Y.T. Liow. Integrating region growind and edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(3):225–233, 1990.

[47] J.H.C. Reiber P.N.J. van der Zwet, G. Koning. Left ventricular contour detection: A fully automated approach. *IEEE Proc. of Computers in Cardiology*, pages 359–362, 1992.

[48] J. Sethian R. Malladi and B. Vemuri. Shape modeling with front propagation: A level set approach. *IEEE Transactions on Pattern Analysis and Machinge Intelligene*, 11(12):1293–1309, 1995.

[49] M. Rivera, O. Ocegueda, and J. L. Marroquin. Entropy controlled Gauss-Markov random measure fields for early vision. *VLSM*, LNCS(3752):137–148, 2005.

[50] Mariano Rivera and Pedro P. Mayorga. Quadratic markovian probability fields for image binary segmentation. *IEEE Workshop in ICV - Reporte Técnico CIMAT*, 2007.

[51] Haralick R.M. Statistical and structural approaches to texture. In *Proceedings of the IEEE*, pages 786–804, 1979.

[52] W.A. Barrett R.R. Stringham and D.C. Taylor. Probabilistic segmentation using edge detection and region growing. In *SPIE Proc. of Visualization in Biomedical Computing*, pages 40–51, 1992.

[53] Mark A. Ruzon. RGB2LAB. *http://ai.stanford.edu/ ruzon/software/rgblab.html*.

[54] D. Geman S. Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984.

[55] A. Sarabi and J.K. Aggarwal. Segmentation of chromatic images. *Pattern Recognition*, 13(6):417–427, 1981.

[56] S. Mori Tamura, H. and Y. Yamawaki. Textural features corresponding to visual perception. *IEEE Transactions on Systems, Man and Cybernetics*, (8):273–247, 1978.

[57] D.C. Taylor and W.A. Barrett. Image segmentation using globally optimal growth in three dimensions with an adaptive feature set. In *SPIE Proc. of Visualization in Biomedical Computing*, pages 98–107, 1994.

[58] Carlo Tomasi. Estimating gaussian mixture densities with em - a tutorial. *http://www.cs.duke.edu/courses/spring04/cps196.1/handouts/EM/tomasiEM.pdf*, 2002.

[59] J.T. Tou and R.C.Gonzalez. *Pattern Recognition Principles*. Addison-Wesley, 1974.

[60] M. Tuceryan and A.K. Jain. Texture segmentation using voronoi polygons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (12):122–216, 1990.

[61] O. Veksler. Image segmentation by nested cuts. In *IEEE Proc. of Computer Vision and Pattern recognition*, pages 18–25, 2000.

[62] Y. Xu and E.C. Uberbacher. 2d image segmentation using minimum spanning trees. *Image and Vision Computing*, 15(1):47–57, 1997.

[63] Carlo Tomasi Yossi Rubner and Leonidas J. Guibas. A metric distributions with applications to image databases. In *Proceedings of the Sixth International Conference on Computer Vision*, pages 59–66, 1998.

[64] Mustafa Özden and Ediz Polat. Image segmentation using color and texture features. In *EUSIPCO*, 2005.