Reporte de Búsqueda, detección y conteo de objetos

Alonso Ramírez-Manzanares
Centro de Investigación en Matemáticas A.C.,
Apdo. Postal 402, Guanajuato, Gto., 36000, Mexico
alram@cimat.mx

1 Introducción

Uno de los principales objetivos de la visión por computadora es poder diferenciar los objetos presentes en una imagen (en este caso, en una imagen digital), de tal manera, que esto logre que la identificación de los mismos (un proceso posterior) sea una tarea mas fácil de realizar. Cuando nos enfocamos en el problema de diferenciar los objetos en cualquier imagen (como es hecho por los seres humanos), una importante pregunta abierta aparece: ¿ Qué información es, tanto suficiente como necesaria, para poder llevar a cabo esta tarea?. Es muy difícil poder expresar este conocimiento o información de una manera algorítmica, por lo que esta respuesta es contestada solamente para algunos casos particulares y no en general.

Dentro de la Visión Computacional, podemos encontrar el procesamiento de imágenes, y dentro de este, una parte muy importante se encarga del análisis de las mismas. Esto es, dada una imagen, lo que deseamos obtener es una descripción de dicha imagen. Los siguientes son ejemplos de problemas de análisis de imágenes:

1. Dado un texto, reconocer las palabras.
2. Dada una imagen aérea de un terreno, clasificar los distintos tipos de suelos (urbano, bosque, lagos, carreteras, etc.)
3. Dada una imagen de un conjunto de células, clasificar las células por tamaño, forma, etc.
4. Dada una imagen médica, detectar tumores, roturas de huesos, etc.

Es decir, dada una imagen, el análisis se encarga de entregar información de ella. Por lo que en todos estos ejemplos, el análisis depende primeramente de detectar determinadas partes de la imagen (regiones u objetos). Para generar tal descripción es necesario segmentar (o separar) adecuadamente e identificar la región deseada.

EL objetivo de este reporte es i) mostrar la consideraciones y problemas que se encuentran en las tareas de distinguir en una imagen digital los objetos de interés de todo lo demás, y ii) ejemplificar algunas de las estrategias o métodos que se han utilizado. Este segundo objetivo es únicamente descriptivo y de ninguna manera trata de mostrar la totalidad de los métodos, ya que simplemente, la enumeración de ellos, es una tarea sumamente difícil, debido a la inmensa literatura existente sobre el tema. Por el contrario, se trata de ejemplificar algunos de los métodos mas famosos dependiendo del tipo de imagen y los problemas que conlleva el poder separar los objetos en la misma.

Este trabajo se estructura como a continuación se explica: La sección 2 define como se ha defindo el objeto y el fondo en una imagen, la sección 3 explica los principales problemas que se encuentran en este tipo de tareas, la sección 4 ejemplifica algunos de los métodos comunmente utlizados en este tipo de tareas, la sección 5 se centra en el tratamiento de las imágenes, una vez que se ha establecido cuáles píxeles corresponden al fondo y cuáles al objeto. Experimentos con varios de los métodos explicados son mostrados en la sección 6 y finalmente las conclusiones son presentadas en la seccion 7.

2 Objetos y Fondo

El separar la imagen en unidades significativas es un paso importante en visión computacional para llegar al reconocimiento de objetos. Este proceso se conoce como segmentación. Una forma de segmentar la imagen es mediante la determinación de los bordes. El dual de este problema, es determinar las regiones; es decir, las partes o segmentos que se puedan considerar como unidades significativas. Esto ayuda a obtener una versión más compacta de la información de bajo nivel, ya que, en vez de miles o millones de pixels, se puede llegar a decenas de regiones, y de ahi reconocer los objetos. Las caraterísticas más comúnes para delimitar o segmentar regiones son: intensidad de los píxeles, textura, color y gradiente. Una suposición importante, que normalmente se asume en visión de nivel intermedio, es considerar que píxeles de un mismo objeto comparten propiedades similares. Por ejemplo, al procesar una imagen de una manzana, suponemos que el color de sus píxeles es aproximadamente homogéneo. En la vida real esto no es totalmente cierto, el color de los píxeles varía. Para evitar este tipo de variaciones es mejor considerar un color "aproximadamente" similar sobre una región mas que a nivel pixel. Esto no es un problema sencillo, ya que es difícil distinguir las variaciones propias del objeto o por cambios de iluminación (por ejemplo, cuando hay sombras en los objetos), de las diferencias por tratarse de otro objeto.

En el análisis de los objetos en imágenes es esencial que podamos distinguir entre los objetos del interés y "el resto", normalmente se hace referencia a este último grupo como el fondo. Las técnicas que se utilizan para encontrar los objetos de interés se refieren generalmente como técnicas de segmentación (segmentar el primer plano del fondo). El resultado de esta segmentación da como resultado una imagen binaria. Normalmente se utiliza la convención de que se le asigan el valor 1 a los píxeles que corresponden a objetos y el valor de 0 a los píxeles que corresponden al fondo. Como resultado de dicha segmentación, la imagen es partida en regiones y se conocen los bordes entre las regiones.

3 Los problemas en la Segmentación de los Objetos y del fondo

Existen diversos métodos para la segmentación reportados tanto en la literatura clásica como en desarrollos recientes. Un hecho es que no existe un solo método que pueda ser aplicado para todos los casos. Dependiendo del problema, es decir, de las características de los objetos y del fondo, será necesario aplicar diferentes métodos. A continuación se explican los casos mas comunes que se encuentran en este tipo de aplicaciones.

3.1 El caso mas simple, objetos y fondo con intensidades de gris homogéneas

Este caso es el mas sencillo y por lo tanto las técnicas de solución son las mas simples e intuitivas. En este tipo de imágenes tanto los objetos como el fondo tienen valores de gris homogéneos y la diferencia entre los valores asociados a cada clase es distinguible. Este tipo de imágenes son raras en la mayoría de

las aplicaciones. Se podría decir que este tipo de imágenes solo se presentan en casos sintéticos y cuando en el proceso de captura el ambiente de iluminación y el proceso de formación de la imagen son altamente controlados, dando como resultado un alto contraste entre el objeto y el fondo. Restricciones que no son fáciles de satisfacer en general. La figura 1a muestra un ejemplo de este tipo de imágenes.

3.2 El problema de *Shading* o variación suave del fondo

Aún dentro del caso en el cual tanto los objetos y el fondo son capturados con intensidades de gris homogéneas, se tiene el problema de que la iluminación no sea homogénea, lo cual conlleva el problema de diferentes intensidades de gris corresponden al mismo objeto. Un ejemplo puede este problema se puede ver en la figura 1b. En este caso no es posible utilizar solamente la información de la intensidad del objeto y se requiere utilizar otras fuentes de información, como por ejemplos los bordes de la imagen. Por otro lado la presencia de ruido en el proceso de captura hace que las intensidades cambién y las zonas homogéneas dejan de serlo.

3.3 Objetos y/o fondo con textura

El caso quizá mas complicado es aquel en el cual los objetos y/o el fondo no tienen como característica espacial que las intensidades son homogéneas y por el contrario su característica es una textura visual. Para entender este concepto, podemos revisar la definición dada por Tamra, Mori y Yamawaky, 1978: "Una región de una imagen tiene una textura constante si el conjunto de estadísticas locales u otras propiedades locales son constantes, varían lentamente o son aproximadamente periódicas". Cabe aclarar, que esta no es la única definición de textura y más grave aún, no existe una definición con la que todos los expertos coincidan o que se pueda aplicar a todos los casos. La figura 1c muestra un ejemplo de este tipo de imágenes. En este caso no es posible clasificar cada pixel de manera independiente, ya que la textura está difinida por regiones de píxeles y no por píxeles individuales. En la práctica siempre es mucha ayuda saber cuantos tipos de textura pueden existir en la imagen, si por ejemplo, todos los objetos tienen la misma textura y grupos de objetos tienen texturas diferentes. Normalmente es necesario formar un vector de razgos o características para cada pixel r y se pueden obtener características de acuerdo al comportamiento del entorno o vecindad de dicho punto. Juzlesz 1973, ha estudiado y definido:

1. Estadísticas de primero Orden. Mide la verosimilitud de observar un nivel de gris en una posición escogida aleatoriamente en la imagen. No tiene dependencia de la interacción entre píxeles

2. Estadísticas de segundo Orden. Están definidas como la verosimilitud de observar un par de valores de gris en los extremos de un dipolo (en este contexto, un dipolo se refiere una estructura de línea que tiene un distancia y una orientación y donde cada uno de sus extremos esta colocado en un pixel de la imagen) en el que tanto el tamaño, la orientación y la posición son seleccionados de manera aleatoria.

En este caso, dado que el concepto de textura se aplica a regiones, es necesario utilizar la información del entorno, es decir, se aplican métodos que utilizan las estadísticas de segundo orden. Es necesario entonces obtener rasgos de los píxeles de la imagen, rasgos que sean los mismos para regiones que tienen la misma textura. La adquisición de este tipo de rasgos seá un de los temas de las siguientes secciones.

a) b) c)

Figure 1: Diferentes tipo de imágenes que conllevan diversos problemas en la detección de los objetos.

En general, podemos concluir que la separación de objetos en una imagen se basa en la detección de fronteras entre partes de dicha imagen, donde la cada parte esta definida por diferentes propiedades, las cuales como se vió anteriormente pueden ser: intensidad de valor de gris o textura.

4 Métodos de clasificación Objeto-Fondo

En las siguientes subsecciones se muestran varios metódos que se utilizan para poder clasificar si un pixel es del tipo objeto o del tipo fondo. Es importante decir que el número de métodos que se pueden encontrar hoy en día en la literatura es exesivamente extenso como para intentar cubrirlos todos. El objetivo de este trabajo es ejemplificar algunos de los métodos utilizados comunmente. En las siguientes secciones se muestran algunos de dichos métodos.

4.1 Segmentación basada en píxeles, umbralizado

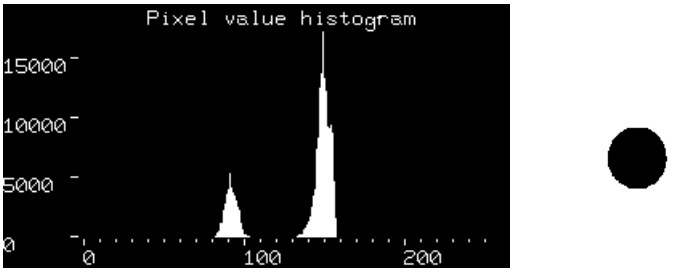
Este es el método de segmentación que conceptualmente es el mas simple. En este metodología, la clasificación se hace pixel a pixel y sin tomar en cuenta las interacciones espaciales de los mismos.

Los procedimientos básicos de este tipo de segmentación son el umbralizado (Thresholding) y el umbralizado basado en el histograma (Histogramming).

El método de umbralizado se aplica a imágenes en las cuales tanto los objetos como el fondo tienen valores homogéneos de gris como los mostrados en la figura 1a. El usuario escoje un valor de intensidad dentro del rango dinámico de la imagen llamado *umbral*, luego, los píxeles con un valor mayor a este umbral son considerados de una clase (objeto o fondo), mientras que todos los demas píxeles son considerados de la otra clase. Esto es, dado un umbral θ, (si los objetos son mas claros que el fondo) los píxeles que conforman a los objetos son definidos como:

$$\{< x, y >\} : I_{x,y} > \theta \tag{1}$$

Si el valor del umbral es determinado tomando en cuenta la información del histograma, el método se conoce como umbralizado basado en el histograma (Histogramming). En este caso el valor del umbral se escoje, tal que este valor divide las dos modas de la distribución del histograma. Los resultados de utilizar esta técnica sobre la figura 1a utilizando el histograma asociado Figura 2a, se pueden apreciar en la figura 2b. Aunque claro está que esta situación ideal rara vez ocurre. Un claro problema es que por lo general



a) b)

Figure 2: El proceso de histogramming.

existe una gran cantidad de píxeles que tienen valores de gris intermedios entre la media de la intensidad de los objetos y la media de la intensidad del fondo (por ejemplo en los bordes). Por lo tanto, la selección de este umbral afecta el tamao de los objetos clasificados en la imagen. Un conocimiento a priori sobre el tipo de frontera (por ejemplo si el borde es simétrico) entre el objeto y el fondo puede ayudar a determinar el umbral correcto, lo cual, por supuesto en muchas aplicaciones no es posible conocer de antemano.

En implementaciones mas sofisticadas, varios umbrales puedes ser especificados si existen varias bandas de información, como en el caso de imágenes multi-espectrales o las bien conocidas imágenes a color RGB.

4.1.1 Corrección del problema de Shading

No obstante que por lo general, los métodos de segmentación basada en píxeles se usan en casos cuando la iluminación es homogénea, aún cuando esta restricción no se cumple, es posible utilizar este tipo de técnicas y obtener resultados aceptables, siempre y cuando se haga un pre–procesamiento que corrija la iluminación. El problema radica en el hecho de que para realizar dicha corrección con exactitud es necesario conocer la naturaleza de la distorción, es decir se necesitan imágenes de referencia adecuadas. Una imagen de referencia adecuada puede ser una toma con un fondo homogéneo (sin objetos) en el cual se pueda apreciar el gradiente del cambio de la iluminación. Una vez que se tiene esta información es posible encontrar las intensidades que se necesitan restar para contrarestar este pernicioso efecto [ver secc 8.3.2 [3]]

Otra manera de poder encontrar un aproximación de la iluminación es filtrar la imagen por un filtro pasabajas, ya que se asume que la iluminación tiene frecuencias mucho mas bajas que las frecuencias de los objetos. Una vez que se tiene la imagen filtrada, esta se resta a la imagen original anulando el efecto del shading.

4.2 Segmentación basada en Regiones

Estas técnicas surgen bajo uno de los supuestos mas importantes en cuanto a la estructura de los objetos en una imagen, la *conectividad*, evitando errores tales como clasificar píxeles aislados como objetos.

En este punto es útil dar la definición de un objeto en una imagen: i) un objeto se define como el conjunto de todos los píxeles que comparten características (intensidad, textura, etc.) y que estan conectados, o bien, ii) un objeto es el conjunto de píxeles que se localizan al interior de una curva cerrada o curva de borde.

4.2.1 Crecimiento de regiones (Region Growing)

Este método se puede explicar por los siguientes pasos básicos

1. Se necesita tener un umbral alto T_H y un umbral bajo T_L dentro del rango dinámico de la imagen.
2. Definir la conectividad. En imágenes 2D se usa vecindad de primer grado con 4 vecinos o bien vecindad de segundo grado con 8 vecinos.
3. Comenzar con un punto de inicio o *punto semilla*, el cual tiene un valor de intensidad arriba del umbral alto. A continuación checar para cada punto vecino si está dentro del rango válido de intensidades, si es así, se agrega a la lista de los píxeles válidos.
4. Escojer un pixel válido como el siguiente y repetir.

La principal desventaja de este método es que muy pequeñas conexiones entre regiones diferentes pueden ocasionar que 2 de ellas se colapsen en una sola. Nuevamente, la manera de escojer el valor de los 2 umbrales no esta establecida.

4.2.2 Algoritmo de división y unión de Regiones (Split and Merge)

En este caso (ver [3]), la partición de la imagen en regiones se desarrolla de acuerdo a un criterio de homogeneidad calculado para cada región. Como medidadas de homogeneidad se pueden establecer umbrales para las medias y las desviaciones de estandar de los valores de gris de las regiones. Es decir se evalua si la media de una región es parecida a la de la otra, o bien la diferencia es menor a un umbral y usando el mismo criterio de diferencia para las desviaciones estandar.

Las 2 operaciones principales que modifican la partición de la imagen son la *división* de una región y la *unión* de 2 regiones. Una región que tiene un valor de homegeneidad mayor a un umbral definido, es subdividida en subregiones. Dos regiones adyacentes pueden ser unidas para generar una nueva región que tiene un valor de homogeneidad menor a un cierto umbral. Se pueden distinguir 3 variantes principales:

1. La estrategia *Top-Down* la cual comienza con una subpartición de la imagen y se incrementa el número de regiones usando el algoritmo de divisiones.
2. La estrategia *Bottom-Up* en la cual se comienza con una sobrepartición de la imagen y se decrementa el número de regines con el algoritmo de uniones.
3. La estrategia híbrida en la cual se modifica la partición actual combinando los algoritmos de división y unión.

En un algoritmo puro de división, se puede partir del hecho de que la imagen es en si es el objeto, se divide en 4 y se mide la homogeneidad. En la contraparte en un algoritmo de unión de regiones, se puede empezar uniendo píxeles vecinos que cumplan con el criterio de homogeneidad.

4.2.3 Algoritmo de conexión en Pirámide (Pyramid Linking)

Burt [10] propuso un algoritmo de conexión en pirámide como una implementación efectiva en un algoritmo de segmerntación de características. En una imagen la característica que se evalua es el nivel de gris de los píxeles.

El algoritmo contiene los siguientes pasos:

1. *Cálculo de una piramide Gaussiana.* Los valores de 4 vecinos son promediados para formar el valor de un pixel en el siguiente nivel mas alto de la pirámide. El nivel inmediato superior de la pirámide tiene la mitad de píxeles que el nivel actual. ver figura 3a.

2. *Segmentación por medio de la conexión en pirámide.* Por la forma de crear la pirámide, cada pixel contribuye a 2 píxeles en el nivel inmediato superior, para decidir a cual pertenece, simplemente se escoje aquel que tiene el valor mas parecido. Esta decisión genera conexiones entre píxeles que se localizan en distintos niveles de la pirámide, formándose una estructura de árbol como se muestra en la figura 3b.

3. *Promediado de los píxeles conectados.* A continuación la estructura de conexión resultante es usada para recualcular la media de los valores de gris, ahora usando unicamente los píxeles que están conectados. Es decir, el valor de cada nodo padre en el arbol es calculado como el promedio de los hijos que estan conectados (ver figura 3c).

4. Los últimos 2 pasos son repetidos iterativamente hasta que el algoritmo converge a una solución, por ejemplo, en la figura 3c se puede ver que el algoritmo convergerá a dos regiones en el nivel penúltimo de la pirámide, ya que en el nivel inmediato inferior las diferencias entre los píxeles son despreciables.

Dentro de las ventajas de este método, podemos mencionar de que no es necesario establecer de antemano cuantas regiones se esta buscando, y que la forma de las regiones no tiene ninguna restricción.

4.2.4 Umbralizado adaptivo (Adaptive Thresholding)

En este caso - a diferencia del método de umbralizado convencional explicado anteriormente - un umbral debe de ser calculado para cada pixel de la imagen.

Existen 2 estrategias principales para determinar dicho umbral [12]. La estrategia de *Chow y Kanenko* y el *umbralizado local.* Ambos métodos están basados en que se asume que es mas probable que la iluminación sea aproximadamente uniforme en pequeñas regiones.

1. *Chow y Kanenko* dividen la imagen en un arreglo de subimagenes que se traslapan y entonces encuentran el umbral óptimo para cada subimagen por medio del análisis del histograma. El umbral para cada pixel es encontrado por medio de la interpolación de los resultados de las subimágenes en las que tiene relación espacial. La desventaja de este método es que es computacionalmente caro y para aplicaciones en robótica de tiempo real es inapropiado.

2. La otra alternativa es encontrar el valor del *umbral local* por medio del análisis estadístico de los valores de intensidad en una vecindad de cada pixel. El estadístico óptimo depende completamente del tipo de imagen que se está anlizando. Los estadísticos mas simples y rápidos de calcular incluyen la media de los valores $T_r = media$, la mediana $T_r = mediana$, o la media de los valores máximos

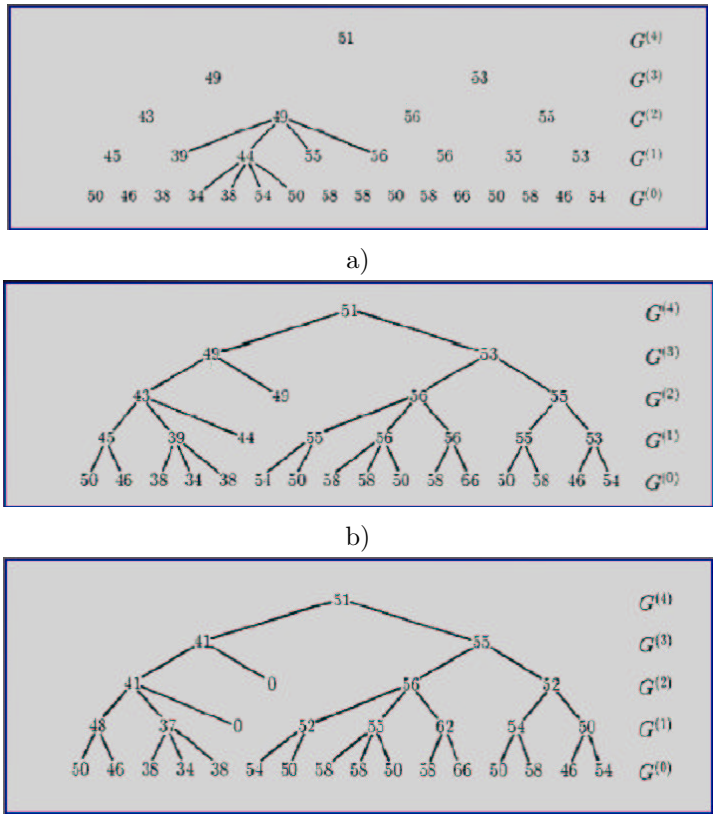


Figure 3: Pasos del método de conexión en pirámide.

y minímos $T_r = \frac{max-min}{2}$ (es importante hacer notar que en el caso cuando existen valores atípicos (outliers) en una región, la mediana es mas robusta que la media, ya que la media es dezplazada significativamente por un solo valor atípico y la mediana no). El tamaño de la vecindad debe de ser lo suficientemente grande para contener suficientes valores de fondo y de objeto, ya que de lo contrario un umbral inadecuado es escogido. En la contraparte, el escoger regiones muy grandes puede violar la suposición de que la iluminación es aproximadamente uniforme. Los puntos anteriores representan las desventajas del método

4.2.5 Método de Watersheds

Una región "watershed" (linea divisoria de las aguas) o bien una cuenca es defina como una región sobre la cual todos los puntos fluyen cuesta abajo hacia el mismo punto en común. Inspirados en geología, uno debería de considerar la región local donde toda el agua de lluvia fluje al mismo lago o río. Esto parecería no ser aplicable a las imágenes con valores de intensidad, pero toma sentido si se aplica a imágenes de gradiente.

Una manera de encontrar *watersheds* es pensar en llenarlos desde abajo hacía arriba y hallar cuáles piscinas se generan. Supóngase que el rango de la imagen es [0,255]. Empezamos primero con los píxeles con valor 0. Claramente no hay ningún pixel con valor menor a ello, por lo tanto forman la base de nuevos *watersheds*. Entonces agregamos todos los píxeles con intensidad 1. Si ellos estan en la vecindad de de

watersheds ya encontradas (con valores de 0), agregamos estos píxeles a estas regiones, de lo contrario creamos una nueva región para estos píxeles.

Este algoritmo se repite para cada valor k hasta el máximo (255 en este ejemplo): cada pixel con intensidad k que es vecino de una región ya etiquetada es agregado a esa región.

Se debe de ser cuidadoso con un detalle: puede haber varios píxeles juntos con intensidad k, y si uno de ellos es adyacente a una región ya etiquetada, todos ellos deben de ser agregados a esta región. Este algorimo básico puede ser implementado de una manera eficiente por medio del uso de estructuras de datos del tipo histograma, las cuales indexan todos los píxeles por su intensidad. Este implementación permite localizar rápidamente todos los píxeles con intensidad k.

Otra posible implentación es denominada *tobogganing*, la cual tiene ese nombre ya que es un proceso análogo a deslizarse por las cordilleras del gradiente hasta llegar a un mínimo local en común. Esta estrategia liga cada pixel p con su vecino de valor menor, este a su vecino de valor menor y así sucesivamente. Se llega aun punto en el cual se alcanza el pixel q el cual es un mínimo local y por lo tanto no tiene vecinos con valores menor a él. Este pixel q es la base de una cuenca. En este momento se establece que apartir de p se alcanzó q con esta estrategia. Todos aquellos píxeles que alcanzan q con esta estrategia pertenecen al mismo *watershed*.

Hay que hace notar que el fondo de la cuenca puede estar conformado por varios píxeles y no solo uno. En este caso, un grupo de píxeles pueden converger a uno de los píxeles de este mínimo y otro grupo de píxeles pueden converger a otro píxel del mismo minímo, con lo cual se detectarían 2 *watersheds* que en realidad son la misma. Es entonces necesario agrupar estos mínimos a una sola región usando un post-procesamiento. El post-procesamiento es simple: al final se tiene que para cada pixel tenemos asociado un identificador $r_i = (x_i, y_i)$, que es la coordenada del pixel a donde se llegó mediante el proceso de descenso, por lo tanto el índice i corre de 1 al número de regiones encontradas hasta este momento. El post–procesamiento investiga uno por uno estos identificadores, digamos que tomamos un r_k y buscamos si existe algún r_j que sea vecino, si esto ocurre (es decir tenemos 2 *watersheds* cuyas bases son vecinas, es decir son la misma), se fusionan las regiones r_k y r_j. Lo anterior se hace hasta convergencia, es decir hasta que ya no es posible fusionar regiones.

4.2.6 Un algoritmo de segmentación EM

Existen algoritmos muy sofisticados para la segmentación de objetos y fondo. Un Algoritmo que es muy utilizado por su buen desempeño es el algoritmo EM (Expectation-Maximization). Como su nombre lo indica este algoritmo consta de 2 pasos, los cuales se ejecutan de manera iterativa hasta alcanzar convergencia. En el paso de *Expectation* se ajusta la mejor segmentación de acuerdo a los parámetros del modelo, mientras que en el paso de *Maximization* los parámetros del modelo son aproximados de acuerdo a la segmentación actual [4].

Supongamos entonces que los objetos y el fondo tienen un valor de intensidad de gris μ_o y μ_f respectivamente, pero que las intensidades han cambiado por la introducción de un ruido Gaussiano (se asume la ditribución Gaussiana por simplicidad en los cálculos, aunque quizá no es la mejor distribución), cambiando su valor de intensidad. Entonces los valores de gris observados vienen de distribuciones:

$$Y_o \sim \mathcal{N}(\mu_o, \sigma_o^2) \tag{2}$$

$$Y_f \sim \mathcal{N}(\mu_f, \sigma_f^2) \tag{3}$$

por lo tanto, la probabilidad de que un pixel en la posición r pertenezca al fondo esta dado por:

$$P\left(I(r)=\mu_f\right)=\frac{1}{\sqrt{(2\pi)}\sigma_f}\exp\frac{-(\mu_f-I(r))^2}{2\sigma_f^2} \tag{4}$$

y la probabilidad de que pertenezca al objeto esta dada por:

$$P\left(I(r)=\mu_o\right)=\frac{1}{\sqrt{(2\pi)}\sigma_o}\exp\frac{-(\mu_o-I(r))^2}{2\sigma_o^2} \tag{5}$$

Estas probabilidades son conocidas como verosimilitudes, ya que parten de la imagen observada. Por lo tanto, los parámetros desconocidos que hay que determinar son la medias y las desviaciones estandar de ambas distribuciones, ya que dadas podemos estimar una verosimilitud para cada pixel de si pertenece al fondo o al objeto.

Además de lo anterior, se asume que tanto el fondo como los objetos son regiones, y no píxeles aislados, por lo que deben de existir zonas homogéneas. Esta homogeneidad se modela en base a campos aleatorios Markovianos [5]. En este enfoque la probabilidad a priori se modela, dando probabilidades altas a configuraciones en donde los píxeles tienen vecinos con valores iguales y probabilidades bajas al caso opuesto. En base a lo anterior, se usa un Gibs Sampler [6] para muestrear la probabilidad de que la imagen modelada como un campo aleatorio Markoviano pertenece a un configuración en base al comportamiento de cada pixel con respecto a su vecindad y a la verosimilitud tomada de los modelos (4) y (5) . Una vez que se tiene la probabilidad muestreada para cada pixel, se obtiene una segmentación preliminar y se recalculan los estimadores de los parámetros de estas distribuciones de la siguiente manera:

$$\widehat{\mu_o}=\frac{\sum_r I(r)*b_o(r)}{\sum_r b_o(r)} \tag{6}$$

$$\widehat{\sigma_o}=\sqrt{\frac{\sum_r (I(r)-\widehat{\mu_o})^2 b_o(r)}{\sum_r b_o(r)}} \tag{7}$$

y

$$\widehat{\mu_f}=\frac{\sum_r I(r)*b_f(r)}{\sum_r b_f(r)} \tag{8}$$

$$\widehat{\sigma_f}=\sqrt{\frac{\sum_r (I(r)-\widehat{\mu_f})^2 b_f(r)}{\sum_r b_f(r)}} \tag{9}$$

donde $b_o(r)$ y $b_f(r)$ son unas funciones indicadoras:

$$b_o(r)=\begin{cases}1 & \text{si r pertenece al objeto}\\ 0 & \text{en caso contrario}\end{cases} \tag{10}$$

$$b_f(r)=\begin{cases}1 & \text{si r pertenece al fondo}\\ 0 & \text{en caso contrario}\end{cases} \tag{11}$$

El cálculo de los estimadores proviene directamente de la definición de media y varianza. La media de los píxeles que pertenecen al objeto, por ejemplo es la suma de los valores de gris de los píxeles que fueron catalogados como pertenecientes al objeto entre el número de píxeles que pertenecen al objeto.

Las funciones indicadoras se obtienen como el resultado del proceso del Gibbs Sampler: si un pixel s tiene probabilidad mas alta de pertenecer al objeto, entonces $b_o(s) = 1$ y $b_f(s) = 0$.

Una vez que se han calculado los nuevos estimadores de parámetros $\widehat{\mu_o}$, $\widehat{\sigma_o}$, $\widehat{\mu_f}$ y $\widehat{\sigma_y}$ se procede nuevamente a el paso *Expectation* para obtener una nueva segmentación y después *Maximization* hasta convergencia.

El principio del algoritmo los estimadores de parmetros son propuestos por el usuario en base a su conocimiento a priori del problema.

El desempeño de este algoritmo es muy bueno, aunque tiene la desventaja de ser costoso computacionalmente.

4.3 Segmentación basada en Bordes

Como se había mencionado en la sección 2, un problema muy importante en la segmentación de los objetos es el problema de *Shading* o iluminación no uniforme, [ver figura 1c]. Aunado a ello, aún con iluminación perfecta, si tenemos 2 clases de objetos unos obscuros y otros claros, al momento de utilizar un umbral, los objetos claros tienden a engrandecerse y los objetos oscuros tienden a empequeñerse. Este problema puede solucionarse si se utiliza un valor umbral para cada objeto, lo cual en la práctica es difícil de determinar.

La segmentación basada en bordes esta basada en el hecho de que la posición de un borde es dada por un máximo en las derivadas de primer orden o un cruce por cero en las derivadas de segundo orden. Por lo tanto, lo que se debe de hacer es buscar los máximos locales en la intensidad del borde. Se puede probar que esta estrategia no está influenciada por las variaciones en la iluminación.

La estrategia típica es:

1. Realizar un barrido de la imagen línea por línea en busca de un máximo local en la magnitud del gradiente.

2. Cuando un máximo es encontrado, se ejecuta un algoritmo de rastreo (*Tracing algorithm*) el cual intenta seguir el máximo del gradiente alrededor del objeto hasta encontrar nuevamente el punto de inicio.

3. Se busca nuevamente un punto de inicio.

4.3.1 Detector de Bordes de Canny

El algoritmo mas utilizado es el de Canny [11], ya que es el detector de bordes ms poderoso que existe actualmente. A continuación se describen los pasos de dicho algoritmo.

El primer paso es el cálculo de la magnitud del gradiente y de la direción del mismo. Nunca se puede esperar que algún algoritmo de detección de contornos trabaje bien con imágenes crudas, por lo tanto, se convoluciona la imagen f con filtros gaussianos uni-dimensionales. De esta forma la imagen se suaviza (eliminacin de ruidos). Aunque la imagen de salida es mas borrosa se gana que un pixel de ruido tiene un efecto pernicioso mas pequeño sobre el cálculo de las derivadas.

A continuación se calcula el gradiente de la imagen suavizada, definido como:

$$\nabla f(x, y) = (\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}) \tag{12}$$

donde a su vez las derivadas son aproximadas por diferencias finitas atrasadas

$$\frac{\partial f}{\partial x} = f(x,y) - f(x-1,y) \tag{13}$$

y se procede a calcular la magnitud del gradiente:

$$|\nabla f(x,y)| = \sqrt{\frac{\partial f}{\partial x}^2 + \frac{\partial f}{\partial y}^2} \tag{14}$$

y la dirección del gradiente:

$$\Theta(x,y) = atan\left(\frac{\frac{\partial f}{\partial y}}{\frac{\partial f}{\partial x}}\right) \tag{15}$$

En este momento tenemos una imagen con las magnitudes del gradiente y una imagen con la dirección del gradiente con cada punto.

El segundo paso adelgazar los bordes, es decir identificar unicamente los puntos que pertenecen al centro del borde y eliminar todos los demas, eso se logra aplicando un algoritmo de *supresión de los no máximos* (non maximum suppression), el cual consiste en poner un cero en todos aquellos puntos que tienen un pixel vecino en la dirección local del gradiente con un valor de magnitud de gradiente mayor a él.

El tercer paso es la ejecución de un algoritmo de rastreo de borde. Canny utilizó un umbralizado con histéresis, como a continuación se explica. Se requieren dos umbrales, uno *alto* y uno *bajo*. Haciendo la suposición de que los bordes importantes deben de estar sobre lineas continuas, nos permite detectar un borde aún en una parte de la imagen donde la linea no esta bien definida (y la magnitud del gradiente es pequeña), pero al mismo tiempo evita que se confundan bordes con píxeles de ruido aislados (los cuales también tienen una magnitud pequeña). Esto marca claramente cuales pueden ser bordes genuinos y cuales no. El proceso es el siguiente: se define un punto de borde como aquel punto que tiene un valor de magnitud mas grande que el umbral alto. Para estos puntos de borde se rastrea si este pixel está conectado a un punto que tenga una magnitud arriba del umbral bajo, si es asi, este punto también se marca como un punto del borde. Este proceso continúa hasta que se llega al punto de donde se partiío o bien ya no hay mas píxeles a lo largo del borde que cumplan la condición de estar arriba del umbral bajo. De esta manera todos los píxeles que tienen una magnitud de gradiente arriba de del umbral bajo, pero que necesariamente estan conectados con puntos con alta confianza en el borde, son marcados como borde, pero puntos *candidatos* a borde que no están conectados son desechados.

Por lo tanto el algoritmo de Canny tiene 3 parámetros, los cuales determinan el funcionamiento del mismo, El tamaño del filtro Gaussiano de suavizado y los dos umbrales.

4.3.2 Diferentes esquemas de cálculo de gradientes

Aunque en la sección anterior se propuso el cálculo de las derivadas parciales en base a diferencias finitas atrasadas, existen varios esquemas para el cálculo de las mismas. Se pueden usa difencias finitas adelantadas:

$$\frac{\partial f}{\partial x} = f(x+1,y) - f(x,y) \tag{16}$$

o bien diferencias finitas centrales:

$$\frac{\partial f}{\partial x} = \frac{f(x+1,y) - f(x-2,y)}{2} \tag{17}$$

cada una de estas aproximaciones tiene sus ventajas y sus desventajas: las diferencias finitas adelantadas y atrasadas no indican el valor discreto de la derivada en el punto x, sino 0.5 de unidad atras o adelante respectivamente. Por otro lado las diferencias finitas centrales no tienen ese problema, pero adolecen del hecho de que el cálculo de la derivada en un punto x no toma en cuenta el valor de la función $f(x)$ en el punto x. Aún con estas desventajas las diferencias finitas son altamente utilizadas como un aproximados de la derivada de un función discreta, ya que los resultados son bastante aceptables y el costo computacional del cálculo es pequeño.

4.3.3 Detector de bordes basado en el operador de Laplace

Uno de los detectores de borde mas usados es el operador de Lapace. Se deriva a partir de las derivadas de segundo orden, ya que en una función 2D, $f(x, y)$

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial f^2}{\partial y^2} \tag{18}$$

En la versión discreta, tenemos que:

$$\frac{\partial^2 f}{\partial x^2} = f(x+1,y) + f(x-1,y) + 2f(x,y) \tag{19}$$

$$\frac{\partial f^2}{\partial y^2} = f(x,y+1) + f(x,y-1) + 2f(x,y) \tag{20}$$

lo que da como resultado

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y) \tag{21}$$

La máscara del filtro usada para la convolución es entonces:

$$\mathbf{L}_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{22}$$

La idea intuitiva es la siguiente, si el coeficiente del centro es positivo los coeficientes alrededor del centro deben de ser negativos y de esta manera la suma de los coeficientes es cero. Es decir el resultado de la convolución de esta mascara es cero cuando existe un cambio de sigo en la imagen.

La mascara anterior nos da un laplaciano isotrópico a 90^0, si se desea tener un Laplaciano isotrópico a 45^o se puede utilizar:

$$\mathbf{L}_2 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{23}$$

Ademas, es común encontrar que se utilizan las máscaras anteriores con los signos de los coeficientes cambiados (positivo en el centro y negativo en los alrededores).

Una ventaja de este detector de bordes es que los bordes detectados no dependen de la rotación de los mismos (a diferencia de otros como es el caso del detector de Sobel [3]) y se usa una sola máscara para detectar todos los bordes.

Las desventajas es que produce bordes dobles, es sensible al ruido (como cualquier derivada de segundo orden) y no es posible detectar la dirección del borde apartir de la imagen de resultado.

4.3.4 El filtro LoG

Los filtros de Laplace tienden a aumentar el nivel de ruido considerablemente, por lo tanto un es posible obtener un mejor detector de bordes si primero se suaviza la imagen con un filtrado Gaussiano y despues se aplica el filtro de Laplace. Esto lleva a un tipo de de detector de bordes regularizado y a una clase de filtros llamados *Laplace of Gaussian* o *LoG*. Es posible calcular la expresión analítica de la derivación del kernel gaussiano y aplicar este kernel en un solo paso, lo cual es computacionalmente más eficiente que convolver 2 veces, una para el suvizado y la otra para el laplaciano.

4.4 Segmentación basada en Textura

Cuando no es posible binarizar tomando en cuenta la homogeneidad de valores de intensidad ya sea en los objetos o en el fondo, es necesario utilizar métodos de segmentación que tomen en cuenta la textura de los mismos. En general es difícil encontrar un solo descriptor de la textura, ya que existen varios problemas asociados a ellos: El detector perfecto debería de ser insensible a rotaciones, a escalamientos o a transformaciones proyectivas. Por lo anterior lo mas común es que se forme una imagen multi–paramétrica. Donde cada imagen contiene un descriptor de los que se verán en las secciones posteriores, mas aparte algun descriptor de la textura basado siempre en una vecindad, ya que la textura se define para regiones y no para píxeles individuales. En fases de procesamiento posteriores, como se verá mas adelante, se podra usar alguno de los varios mtodos para detectar clusters n-dimensionales para separar las regiones.

Cuando se calculan las características de un punto locales en base a su vecindad, normalmente se usa una ventana (Gaussiana) para limitar la contribución de la información conforme la distancia al punto en cuestión aumenta.

Los rasgos o característica de textura son coeficientes que texturas que son iguales para texturas iguales. A continuación se presentan algunos métodos de detección de rasgos o características de textura.

4.4.1 Matrices de co-ocurrencia

Los estimadores espaciales de co-ocurrencia de niveles de gris, estiman propiedades de las imágenes relacionadas con los estadísticos de segundo orden.

La $G \times G$ matriz de co-ocurrencia de niveles de gris $P_{\mathbf{d}} = (dx, dy)$ para un vector de dezplazamiento $\mathbf{d}$ esta definida formalmente como:

$$P_{\mathbf{d}}(ij) = |\{((r,s),(t,v)) : I(r,s) = i, I(t,v) = j\}| \tag{24}$$

donde $(r,s),(t,v) \in N \times N$, $(t,v) = (r+dx, s+dy)$, y $|\cdot|$ es la cardinalidad del conjunto. Ejemplo: para la imagen I de dimensión 4×4 con 3 niveles de gris, tenemos

$$I = \left|\begin{array}{c|c|c|c} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 2 & 2 \end{array}\right|, P_{\mathbf{d}} = \begin{pmatrix} 4 & 0 & 2 \\ 2 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix}$$

Característica	Fórmula
Energía	$\sum_i \sum_j P^2_{\mathbf{d}}(i,j)$
Entropía	$-\sum_i \sum_j P_{\mathbf{d}}(i,j) \log(P_{\mathbf{d}}(i,j))$
Contraste	$\sum_i \sum_j (i-j)^2 P_{\mathbf{d}}(i,j)$
Homogeneidad	$\sum_i \sum_j \frac{P_{\mathbf{d}}(i,j)}{1+\lvert i-j \rvert}$
Correlación	$\frac{\sum_i \sum_j (i-\mu_x)(j-\mu_y) P_{\mathbf{d}}(i,j)}{\sigma_x \sigma_y}$

Table 1: Características de textura en base a la Matriz de co-ocurrencia

La matriz de co-ocurrencias indica algunas propiedades hacerca de la distribución espacial de niveles de gris en la textura de una imagen. Por ejemplo, si observamos que la diagonal de esta matriz tiene los valores mas grandes, podemos deducir que la textura es burda con respecto al vector de dezplazamiento **d**.

Se han propuesto varias características de la textura que pueden ser calculadas a partir de la matriz de co-ocurrencia. Se define :

$$P_{\mathbf{d}}(x) = \sum_j P_{\mathbf{d}}(x,j) \tag{25}$$

$$P_{\mathbf{d}}(y) = \sum_i P_{\mathbf{d}}(i,y) \tag{26}$$

y definimos μ_x y μ_y como las medias y σ_x y σ_y como las desviaciones estándar de $P_{\mathbf{d}}(x)$ y $P_{\mathbf{d}}(y)$ respectivamente, con lo cual tenemos las características mostradas en la tabla 1. Los nombres de estas características se ha definido así para dar una idea de lo que representan.

Las características basadas en la matriz de co-ocurrencia adolecen de varias dificultades. La primera es que no esta bien defino como escoger el vector de desplazamiento **d**, y calcular varias matrices para diferentes valores de **d** es computacionalmente costoso, ya que para cada uno de ellos existen un gran número de características a evaluar.

4.4.2 Medidas de Textura de Law

La medidas de energía desarrolladas por Kenneth Ivan Laws han sido usadas en diversas aplicaciones. Esas medidas son calculadas en 2 pasos: primero se aplican pequeños kerneles de convolución a la imagen digital y luego se aplica una operación de ventana no lineal.

Los kerneles típicos de de convolución en 2D para discriminación de texturas son generados a partir del siguiente conjunto de kerneles de convolución unidimensional de longitud 5:

```
L5  =  [   1   4  6   4   1  ]
E5  =  [  -1  -2  0   2   1  ]
S5  =  [  -1   0  2   0  -1  ]
W5  =  [  -1   2  0  -2   1  ]
R5  =  [   1  -4  6  -4   1  ]
```

Los nemotécnicos utilizados se refieren a nivel (Level), borde (Edge), mancha (Spot), ondulación (Wave) y rizo (Ripple). Es importante notar que la suma de todos los kernel excepto L5 es igual a cero.

De esos kernels de convolución uni-dimensionales se pueden generar 25 diferentes kerneles de convolución bi-dimensional por medio de la convolución de un kernel vertical con otro horizontal. De los 25 dos-dimensional kernels que se pueden general, 24 de ellos suman cero excepto el L5L5. La lista de todos los 5×5 kerneles generados es:

```
L5L5 E5L5 S5L5 W5L5 R5L5
L5E5 E5E5 S5E5 W5E5 R5E5
L5S5 E5S5 S5S5 W5S5 R5S5
L5W5 E5W5 S5W5 W5W5 R5W5
L5R5 E5R5 S5R5 W5R5 R5R5
```

La manera genérica de utilizar estos kerneles se explica a continuación:

1. Aplicar los kerneles de Convolución. Dada una imagen de ejemplo a la que se le deasea aplicar un análisis texturas (calcular características de textura en cada pixel), primero se aplican cada uno de los 25 kerneles de convolución a dicha imagen (por supuesto, para algunas aplicaciones solo es necesario usar un subconjunto de ellos). El resultado es un conjunto de 25 $N \times M$ imágenes en escala de gris, las cuales componen la base de este análisis.

2. Aplicar una operación de ventana. En este paso vamos a reemplazar cada pixel de nuestras 25 $N \times M$ imágenes en escala de grises por una "Medida de Energía de la Textura" (TEM). Esto se hace tomando en cuenta una vecindad local alrededor de cada pixel y sumando todos los valores absolutos de los píxeles vecinos. Con lo anterior se ha generado otro conjunto de imágenes las cuales son nombradas como el conjunto TEM de imágenes. Esto es, el siguiente filtro no-lineal es aplicado a las 25 imágenes:

$$New(x, y) = \sum_{i=-l}^{l} \sum_{j=-l}^{l} |Old(x + i, y + j)| \tag{27}$$

   Laws sugiere que se puede aplicar otro filtro diferente al absuluto, el cual es presentado a continuación:

$$New(x, y) = \sqrt{\sum_{i=-l}^{l} \sum_{j=-l}^{l} Old(x + i, y + j)^2} \tag{28}$$

   En este punto se han generado 25 imágenes a partir de la original. Estas imágenes van a ser nombradas de acuerdo al nombre del kernel aplicado con una "T" al final, que indica que es una medida de "energía" de la textura, ya que un filtrado no lineal ha sido aplicado. Las imagens TEM son las siguientes:

```
L5L5T E5L5T S5L5T W5L5T R5L5T
L5E5T E5E5T S5E5T W5E5T R5E5T
L5S5T E5S5T S5S5T W5S5T R5S5T
L5W5T E5W5T S5W5T W5W5T R5W5T
L5R5T E5R5T S5R5T W5R5T R5R5T
```

3. Normalizar las características en contraste. Todos los kerneles que hemos utilizado tienen una media zero excepto el L5L5. De acuerdo a la sugerencia de Laws, es posible utilizar este como una imagen de normalización; si normalizamos las imágenes TEM pixel por pixel con el valor de la imagen L5L5T, esto normalizará las características para el contraste.

4. Combinación de características similares. Para muchas aplicaciones, la orientación de las texturas no es importante. Si este es el caso, entonces características similares pueden ser combinadas para remover el sesgo en la orientación. Por ejemplo L5E5T es sensible a bordes verticales y E5L5T es sensible a bordes horizontales. Si se suman esas 2 imágenes TEM, entonces tenemos una sola característica que es simplemente sensible a bordes.

   De esta manera las características que fueron generadas con kerneles de convolución transpuestos son adicionados juntos. Para identificar estas nuevas imágenes, les agregamso la letra R al final, la cual denota invarianza rotacional.

```
E5L5TR = E5L5T + L5E5T
S5L5TR = S5L5T + L5S5T
W5L5TR = W5L5T + L5W5T
R5L5TR = R5L5T + L5R5T
S5E5TR = S5E5T + E5S5T
W5E5TR = W5E5T + E5W5T
R5E5TR = R5E5T + E5R5T
W5S5TR = W5S5T + S5W5T
R5S5TR = R5S5T + S5R5T
R5W5TR = R5W5T + W5R5T
```

   Para mantener la consistencia de las características con respecto a la contribución, se deben escalar las características restantes por 2:

```
E5E5TR = E5E5T * 2
S5S5TR = S5S5T * 2
W5W5TR = W5W5T * 2
R5R5TR = R5R5T * 2
```

   El resultado (si asumimos que hemos eliminado el kernel L5L5T como fue sugerido en el paso 3), es un conjunto de 14 características de textura que son invariantes rotacionalmente. La apliación de esta 14 imágenes nos da un vector de características de 14 componentes para cada pixel.

4.4.3 Características en base a los mosaicos de Voronoi

Los mosaicos de Voronoi han sido utilizados como generadores de tokens de textura [8], debido a sus propiedades espaciales de definición de la vecindad local y porque la distribución espacial de estos tokens esta reflejada en las formas de los poligonos de Voronoi, ver figura 4.

El algoritmo de para generar los mosaicos de Voronoi, apartir de un serie de puntos puede ser consultado en [7].

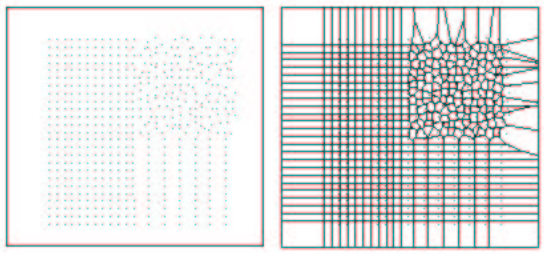


Figure 4: 2 patrones de puntos y los mosaicos de Voronoi asociados.

Para poder aplicar un método geométrico como este a una imágen de valores de gris, es necesario obtener los tokens. Una manera de obtener tokens (o simplemente puntos) en las imágenes de niveles de gris es descrita a continuación:

1. Aplicar filtro del Laplaciano del Gaussiano.
2. Seleccionar aquellos píxeles están en un máximo local en la imagen filtrada. (Se dice que un pixel esta en un máximo local si su valor es mas grande que seis de sus 8 vecinos)
3. Ejecutar un análisis de componentes conectados (ver sección 5.1) en la imagen binaria resultante del paso anterior usando los 8 vecinos mas cercanos. Cada componente conectado define un token.

Una vez que se ha construido los mosaicos de Voronoi, se extraen características de cada celda y tokens que tienen características similares son agrupadas para construir regiones de textura uniforme.

Los momentos de área de los polígonos de Voronoi sirven como un conjunto útil de caracteísticas, ya que reflejan tanto la distribución espacial tanto las formas de los tokens en la imagen texturizada. El momento de area de orden $(p+q)$ de una región cerrada R con respecto aun token con coordenadas (x_0, y_0) es definido como:

$$m_{pq} = \int \int_R (x - x_0)^p (y - y_0)^q dxdy \tag{29}$$

donde $p + q = 0, 1, 2, ...$ (ver sección 5.4.6). Una descripción de las cinco características usadas se dan en la tabla 2 donde $(\overline{x}, \overline{y})$ son las coordenadas del centroide del polígono de Voronoi. En ella se muestran características como f_2, la cual da la magnitud del vector del token al centroide del polígono, f_3 da la dirección, f_4 indica la elongación completa del polígono ($f_4 = 0$ para un círculo) y f_5 indica la orientación del eje mayor.

Estas características han sido utilizadas para la segmentación de texturas. El algoritmo de segmentación esta basado en bordes: una gran diferencia (estadística) entre las características de textura es evidencia de un borde en la textura.

4.4.4 Modelos de Gabor

Una de las formas mas utilizadas para la detección de texturas es el banco de filtros de Gabor [8, 13].

La idea de utilizar un banco de filtros es la siguiente, hacer pasar la imagen f por este conjunto de filtros y obtener una imagen como resultado del proceso con cada uno de los filtros, es decir una imagen r_i. Como a continuación se verá, un filtro entonado a una cierta frecuencia responderá con una alta magnitud donde

Característica de Textura	Expresión
f_1	m_{00}
f_2	$\sqrt{\overline{x}^2 + \overline{y}^2}$
f_3	$atan(\overline{y}/\overline{x})$
f_4	$\frac{\sqrt{(m_{20}-m_{02})^2+4m_{11}^2}}{m_{20}+m_{02}+\sqrt{(m_{20}-m_{02})^2+4m_{11}^2}}$
f_5	$atan(\frac{2m_{11}}{m_{20}-m_{02}})$

Table 2: Características de los mosaicos de Voronoi

exista esa cierta frecuencia y atenuará las zonas en donde no esté presente. Si partimos del hecho de que una textura esta conformada por varias franjas en diferentes orientaciones y a diferentes frecuencias, podemos separar la imagen por regiones donde los filtros indicaron que existían frecuencias diferentes. Esto se explica a detalle a continuación.

Una imagen de entrada $I(x, y)$, $x, y \in \Omega$ (donde Ω es el conjunto de puntos de la imagen) es convolucionada con una función de Gabor bidimensional compleja $g_c(x, y)$, $x, y \in \Omega$, para obtener una imagen compleja de características de Gabor, esto es, la imagen real e imaginaria son:

$$r^{(r)}(x, y) = \int \int I(\xi, \eta) g^{(r)}(x - \xi, y - \eta) d\xi d\eta \tag{30}$$

$$r^{(i)}(x, y) = \int \int I(\xi, \eta) g^{(i)}(x - \xi, y - \eta) d\xi d\eta \tag{31}$$

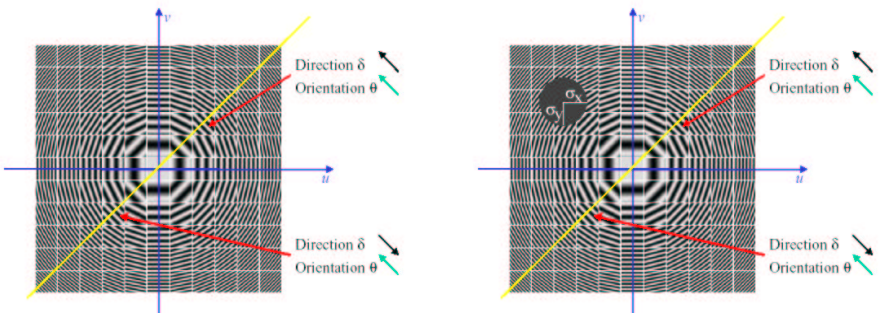
Para entender las funciones bidimensionales de gabor es necesario describir como se visualiza un filtro de gabor en la frecuencia. La imagen 5a muestra como se distribuyen las diferentes frecuencias de una imagen en el dominio de la frecuencia (u, v). Podemos ver que las frecuencias mas altas se encuentran mas alejadas de origen (u,v)=(0,0) y que la orientación de estas frecuencias es perpendicular a la línea de ángulo Θ que une esta frecuencia con el origen de coordenadas. El filtro de gabor, visulaizado en este espacio, y el cual dejará pasar algunas de estas frecencuencias está definido entonces como una ventana Gaussiana con los siguientes 4 parámetros:

1. Los numeros u, v que definen la posición del filtro. Esta posición es claramente también definida por otros dos parámetros: el ángulo Θ y la distancia al origen ρ.

2. El ancho de la ventana gaussiana en la dirección u, denominado σ_u, y el ancho de la ventana Gaussiana en la dirección v, denominado σ_v.

La diferencia fundamental entre un fitro de Gabor y un filtro pasabando radica en el hecho de que como el filtro de gabor no es simetrisco (esta posicionado solo para frecuencias positivas) la respuesta es la envolvente de las frecuncia a donde ha sido entonado, la cual entrega espacialmente. Por ejemplo, supongamos que queremos que nuestro filtro de Gabor asentúe en nuestra imagen las zonas donde hay franjas con frecuencia media y pendiente unitaria, en este caso el filtro de gabor se posiciona como se muestra en la figura 5b.

Una vez que hemos determinado el filtro de Gabor correspondiente mediante la colocación de una función Gaussiana, se procede a calcular la Transformada Inversa de Fourier la cual nos entrega los 2 kerneles de convolución $g^{(r)}(x, y)$ y $g^{(i)}(x, y)$, real e imaginario respectivamente.



a) b)

Figure 5: a) Relación de las franjas y su posición en el espacio de la frecuencia. b) Colocación de un filtro de Gabor.

La magnitud punto a punto de esta imagen $M(x,y) = \sqrt{g^{(r)}(x,y)^2 + g^{(r)}(x,y)^2}$ nos indica en que regiones de la imagen esta presente esta textura.

Es claro que, sin conocimiento a priori del tipo de franjas que contiene una imagen, el número posible de filtros a probar (combinaciones de los 4 paraámetros descritos anteriormente) es muy grande y por lo tanto el proceso completo requiere de un tiempo computacional considerable.

Mediante la utilización del banco de filtros de Gabor, tenemos un vector de características para cada punto de la imagen, donde la entrada i-ésima de este vector es la magnitud ($M_i(x,y)$) obtenida del procesamiento de la imagen con el i-ésimo filtro de Gabor.

4.4.5 Local binary Pattern (LBP)

La primera versión o versión original de este operador trabajaba con los 8 vecinos de cada pixel, usando el valor del pixel central como umbral. Supongamos que estamos posicionados en el pixel r y este tiene los ocho vecinos denominados s_i, i=0,1,..,7. Este umbralizado se realiza de la siguiente manera:

$$s_i^{(b)} = u(s_i - r) \tag{32}$$

con

$$u(x) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases} \tag{33}$$

El codigo LBP es generado por la multiplicación de los valores de los vecinos umbralizados $s_i^{(b)}$ por los pesos asignados a cada vecino. Este proceso se muestra en la figura 6.

Dado que el codigo LBP es por definición invariante a cambios monótonos en la escala de gris, este operador detecta únicamente características de textura y no es alterado por cambios en la iluminación de la imagen.

4.5 Métodos de clasificación basados en los vectores de razgos. Clustering

Para poder separar las diferentes regiones de la imagen basados en un conjunto de caractaerísticas para cada pixel (características como las descritas en las secciones anteriores), se modela este conjunto como un vector

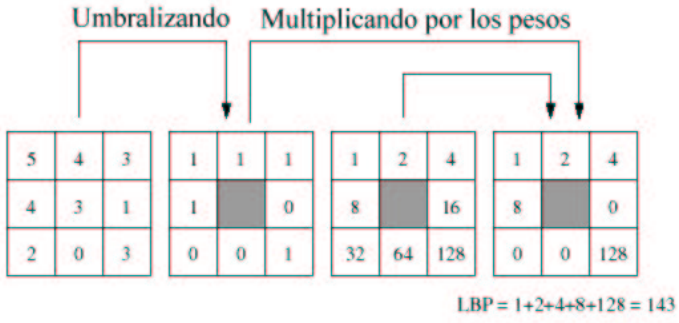


Figure 6: Ejemplo del cálculo del codigo LBP

n-dimensional asociado a cada poisición r de la imagen. Es entonces la tarea detectar cúmulos o clusters en este espacio n-dimensional, donde un cluster indica que esos píxeles comparten carecterísticas o rasgos.

En la literatura se pueden encontrar un número exorbitante referencias de métodos para clusterizar. La clasificacin básica de estos métodos es en metodos a los cuales se les indica de antemano cuantas clases o clusters se desea encontrar y metodos en los que este número es determinado por el mismo algoritmo. En nuestro sencillo caso de segmentación entre fondo u objeto, se asume que solo hay dos clases, por lo que un algoritmo donde se define de antemano el número de clase puede trabajar bien. A manera de demostración en esta sección se describe uno de los métodos mas sencillos e intuitivos de clusterización llamada k-medias.

4.5.1 Agrupamiento por K-medias

Este algoritmo consiste de 3 pasos principales [4]:

1. Inicialización. Consiste en inicializar arbitrariamente los centros de los K grupos, para esto simplemente se puede incializar como los centros con K de los puntos a clusterizar, o bien si se tiene información a priori del comportamiento de los centros, se puede inicializar con esta, lo cual aceleraría la convergencia.

2. Asignación y actualización de los centros. En este paso se asigna cada patrón al grupo ms cercano y se recalculan los centros en base a esta asignación. Aqui conviene hablar de las distancias entre los patrones o vectores: si todas las entradas del vector (o características) tienen el mismo peso, entonces se puede usar la distancia euclidiana, de otra manera se debería de asigna pesos a cada característica de tal forma que las caracteísticas mas relevantes tengan una mayor influencia en el cálculo de la distancia.

3. En el paso 2, algunos patrones pueden cambiar de agrupamiento y en consecuencia, los centros de éstos. Si ésto ocurre, se trata de repetir dicho paso hasta que no se modifiquen los centros. Cuando no hayan modificaciones se considera que se ha encontrado una buena partición y se termina el agrupamiento.

5 Detección y conteo de Objetos en imágenes binarias

Hasta lo que se ha expuesto, tenemos un conjunto de píxeles que pertenecen al fondo y un conjunto que pertenece al objeto. Ahora es necesario agrupar esto píxeles para formar objetos o componentes conectados, ya que por definición un objeto es un conjunto de píxeles que pertenecen a la clase objeto, pero que estan conectados.

Como siguiente paso, se pueden obtener muchas características de dichos objetos, los cuales nos pueden servir para diferenciar objetos de ciertas características, por ejemplo, objetos grandes de peque´os, concavos de convexos, etc. En esta sección se presentan métodos para realizar las dos tareas anteriores.

Dentro de los descriptores de forma de los objetos, se cuenta con una gran variedad: desde los mas sencillos e intuitivos, hasta los mas complicados. Sin embargo, dependiendo de la aplicación, es necesario escojer algun descriptor de forma, es decir, la aplicación definirá el descriptor de forma mas adecuado para una posible clasificación.

5.1 Análisis de componentes conectados y etiquetado

Una vez que tenemos la imagen ya binarizada indicando cuales píxeles pertenecen al fondo y cuales al objeto, es necesario entonces agrupar estos píxeles en objetos o componentes. Es decir, es posible tener 500 píxeles de tipo objeto pero, digamos por ejemplo, solo 3 objetos.

A continuación se presentan algunas ideas para realizar dicho procesamiento.

5.1.1 Búsqueda exhaustiva basada en la conectividad de los píxeles

El algoritmo básico para establecer lo componentes conectados de una imagen binaria asigna una etiqueta diferente (normalmente un numero entero) a cada componente u objeto de la imagen. A continuación se indica el algoritmo básico :

1. Se iniciliazan las etiquetas de todos los píxeles que pertenecen a la clase objeto a -1. Se inicializa un identificador secuencial de etiquetas a C=0. Se inicializa una lista de equivalencia de etiquetas a vacia ($LE = \emptyset$).

2. Se recorre la imagen renglón a renglón de arriba a bajo y de izquierda a derecha, hasta encontrar un pixel de p que no tiene etiqueta asignada, es decir, $E(p) = -1$.

3. Se establecen los 4 posibles vecinos que ya han sido visitados por el algoritmo, y por lo tanto que ya han sido etiquetados. Estos 4 vecinos son : izquierda, arriba, diagonal izquierda-arriba y diagonal derecha-arriba.

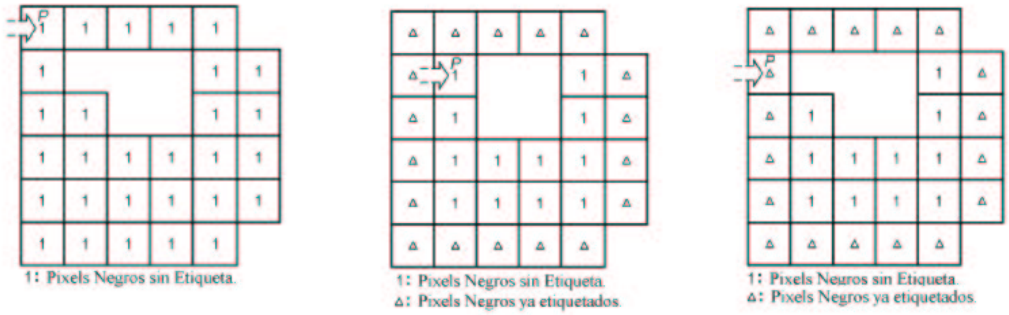
4. Si ninguno de los 4 vecinos tiene una identificador de etiqueta, se asigna la nueva etiqueta actual al punto p ($E(p) = C$) y se incrementa el número de etiqueta actual: $C = C + 1$.

5. Si solo uno de los 4 vecinos q tiene un identificador de etiqueta, entonces asignar esta etiqueta al pixel actual $E(p) = E(q)$.

6. Si hay diferentes etiquetas en los 4 vecinos asignar a p la etiqueta de cualquiera de ellos, digamos la de r. Ya que estas regiones están conectadas tomar nota de la equivalencia, es decir agregar a la lista estas nuevas equivalencias, por ejemplo $EL.add(E(p) == E(q))$.

7. En el momento en que no se encuentran mas puntos para procesar se ha terminado la asignación.

8. Se debe de recorrer la lista de equivalencias y se asigna una nueva etiqueta para todas las etiquetas anteriores que son equivalentes.



a) b) c)

Figure 7: Algoritmo de rastreo de contorno. a) Caso 1 , contorno externo, b) Primera variante del caso de contorno interno, c) Segunda variante del contorno interno. Ver el texto.

9. Se recorre la imagen nuevamente y para cada etiqueta vieja se le asigna la nueva que etiqueta que junta componentes equivalentes.

Este algoritmo es el enfoque básico. Tiene la desventaja que hay que recorrer la imagen 2 veces.

5.1.2 Un Método de etiquetado usando un algoritmo de rastreo de contorno.

La idea de este algoritmo es usar una técnica de rastreo de contorno para detectar los contornos de los objetos y también para rellenar las areas internas [14]. Todos los componentes son etiquetados en un solo barrido de la imagen y se les es asignada una nueva etiqueta o la misma etiqueta de los píxeles vecinos.

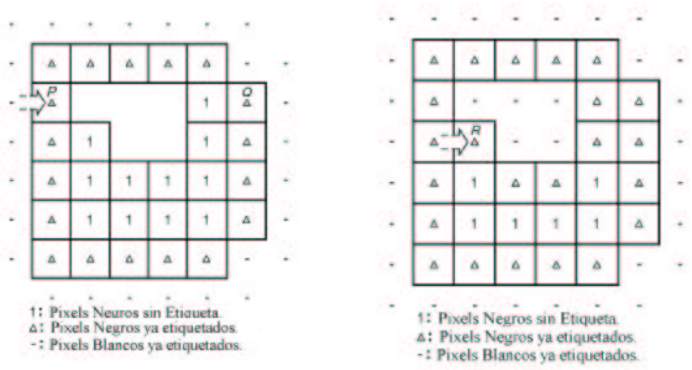
El algoritmo consiste de los siguientes pasos:

Para cada imagen I se le asocia una imagen L en la cual se van a guardar las etiquetas asignadas a todos los componentes. Todos los píxeles de L son inicializados a 0. Sea C el índice de etiquetado, el cual es inicializado a 1.

Se procede a barrer la imagene de arriba a abajo y de izquierda a derecha buscando un pixel de objeto P, el cual denominaremos como un pixel negro (y el fondo blanco). El estado de este pixel puede caer dentro de los 3 siguientes casos:

1. P no tiene etiqueta y su vecino de la izquierda es un pixel blanco (Figura 7a). En este caso P debe ser un punto externo del contorno y se ejecuta el procedimiento de *RASTREO DE CONTORNO* para encontrar el contorno que contiene a P. Se asigna la etiqueta C a todos los píxeles del contorno y se incrementa el valor de C.

2. El vecino de la derecha de P en un pixel blanco no etiquetado. En este caso, P debe de ser un pixel de contorno interno y por lo tanto se ejecuta tambien la rutina *RASTREO DE CONTORNO*. En este caso hay dos posibilidades sobre la etiqueta que le corresponde al contorno. Primera posibilidad: P no tiene etiqueta (figura 7b), en este caso el vecino izquierdo de P ya debió de haber sido etiquetado, por lo tanto se asigna a P y a todo el contorno la etiqueta del vecino izquierdo de P. Segunda posibilidad: P ya ha sido etiquetado (figura 7c) y se le asigna a todo el contorno esta etiqueta. Nótese que es importante diferenciar en este caso el pixel P del tipo de píxeles como el Q en la figura 8a, el cual tiene



a) b)

Figure 8: El uso de etiquetas negativas en el rastreo de contorno. Ver texto.

5	6	7
4	*P*	0
3	2	1

Figure 9: Convención de posiciones relativas a un pixel en el procedimiento *RASTREADOR*

también un vecino a la derecha que es blanco. La manera de evitar ejecutar la rutina de *RASTREO DE CONTORNO* en Q es etiquetando los píxeles que rodean al componente con un número negativo en donde ya se ha trazado un contorno. Por lo tanto, al momento en que Q es alcanzado por la búsqueda line a linea, el vecino de la derecha de Q ya no es un pixel blanco sin etiqueta. Por otro lado, en la figura 8a se puede ver que el vecino de la derecha de P todavia no tiene etiqueta, dado que el contorno interno que contiene a P no ha sido rastreado en el momento que P es encontrado. El etiquetar con numeros negativos los píxeles que rodena a los componentes permite ademas evitar que el contorno sea rastreado mas de una vez apartir de cualquier otro punto del contorno. Por ejemplo en la figura 8b se muestra como el caso 2 no se cumple para R ya que el vecino de la derecha ya tiene una etiqueta negativa.

3. Ninguna de las condiciones anteriores. En este caso el vecino de la izquierda de P ya debe de tener una etiqueta y por lo tanto asignamos a P dicha etiqueta.

A continuación se indica el algoritmo del procedimiento *RASTREO DE CONTORNO*. El objetivo de *RASTREO DE CONTORNO* es encontrar un contorno ya sea interno o externo apartir deun pixel dado, el cual llamaremos S. En este pixel se ejecuta el procedimento *RASTREADOR*. Si *RASTREADOR* identifica a S como un punto aislado se termina el procedimiento de *RASTREO DE CONTORNO*. De otra manera *RASTREADOR* entregaraá el siguiente punto del contorno el cual denominaremos T. Entonces se continúa ejecutando *RASTREADOR* sucesivamente hasta que las dos siguientes condiciones on satisfechas (i) *RASTREADOR* entrega el punto S de nuevo y (ii) el siguiente punto del contorno de S es T de nuevo, con lo cual garantizamos el contorno de objetos que tienen anchura 1.

A continuación se indica el algoritmo del procedimiento *RASTREADOR*. El objetivo de este proced-

imiento es buscar el siguiente punto del contorno apartir de un punto actual P. El rastreo es controlado por la dirección inicial de búsqueda, la cual es determinada por la posición del punto anterior del contorno como a continuación se explica. Para un punto P dado, las posiciones relativas de los vecinos son identificadas con numeros del 0 al 7 como se muestra en la figura 9. Si P es un punto inicial de un contorno externo, la dirección inicial de búsqueda es 0, dado que todos los píxeles de arriba y de la izquierda no pueden pertenecer a este contorno (de otra manera el contorno habría sido encontrado antes). Si de otra manera P es el punto inicial de un contorno interno la poisción inicial de búsqueda es 1, ya que sabemos que el vecino de la derecha en un pixel blanco. Por otro lado cuando buscamos el siguiente punto del contorno la dirección inicial es asignada como $d + (2\ mod\ 8)$, donde d es la posición relativa con respecto a P donde esta localizado el punto anterior del contorno. Lo anterior se deriva del hecho que la posición (d+1 mod 8) necesariamente fue analizada cuando se detectó que P era el siguiente punto del contorno. Una vez que se establece la dirección inicial de búsqueda se procede a buscar el siguiente punto negro (del objeto) en el sentido de las manecillas del reloj. Este punto es el devuelto por *RASTREADOR*. Si se han recorrido todos los vecinos y no se encuentra ningun pixel negro, *RASTREADOR* indica que P es un punto aislado. La tarea de marcar con números negativos los píxeles que rodean un objeto se hace en este procedimiento. Esto se hace de la siguiente manera: en la búsqueda secuencial en el sentido de las manecillas del reloj los puntos que no pertenecen al objeto son etiquetados con un número negativo.

Este algoritmo es eficiente en velocidad ya que etiqueta los componentes en una sola pasada y permite tener una representación de los contorno al mismo tiempo que etiqueta (los píxeles con etiqueta negativa).

5.2 Operadoradores morfológicos

En el contexto de imágenes, el procesado morfológico se refiere al estudio de la topología de las estructuras de los objetos a partir de sus imágenes. Se trata de operaciones sobre objetos binarizados mediante elementos estructurales, que varían la forma de estos de una manera definida.

Cuando se habla de procesado morfológico se habla fundamentalmente de dos operaciones básicas: erosión y dilatación.

5.2.1 Erosión

Si tenemos un objeto $\mathbf{X}$ y un elemento estructural B_x cuyo origen ha sido trasladado a la posición x, la erosión de $\mathbf{X}$ por B_x está definida por todos los puntos de x tales que B_x está incluida en $\mathbf{X}$. Se puede representar formalmente como:

$$\mathbf{X} \ominus B = \{x : B_x \subset \mathbf{X}\} \tag{34}$$

5.2.2 Dilatación

De la misma forma, la dilatación estará formada por todos los puntos x tales que B_x "toca" a $\mathbf{X}$:

$$\mathbf{X} \oplus B = \{x : B_x \cap \mathbf{X} \neq \emptyset\} \tag{35}$$

Ejemplos de erosión y dilatación pueden ser vistos en la figura 10

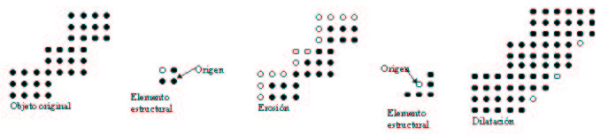


Figure 10: Ejemplos de erosión y dilatación

5.2.3 Propiedades de la erosión y dilatación

- No son inversas una de la otra

$$(\mathbf{X} \oplus B) \ominus B \neq \mathbf{X} \tag{36}$$

- Distributividad

$$(\mathbf{X} \oplus (B \cup C) = (\mathbf{X} \oplus B) \cup (\mathbf{X} \oplus C)) \tag{37}$$

$$(\mathbf{X} \ominus (B \cup C) = (\mathbf{X} \ominus B) \cap (\mathbf{X} \ominus C)) \tag{38}$$

- De conocimiento total

$$(\mathbf{X} \cap \mathbf{Z}) \ominus B = (\mathbf{X} \ominus B) \cap (\mathbf{Z} \ominus B) \tag{39}$$

- Iteración

$$(\mathbf{X} \ominus B) \ominus C = \mathbf{X} \ominus (B \oplus C) \tag{40}$$

$$(\mathbf{X} \oplus B) \oplus C = \mathbf{X} \oplus (B \oplus C) \tag{41}$$

- Dualidad

$$\mathbf{X}^c \oplus B = (\mathbf{X} \ominus B)^c \tag{42}$$

Basados en estos operadores bśicos se han definido dos nuevos operadores llamados *cerradura* y *apertura*. La cerradura se define como la aplicación de una erosión y a continuación una dilatación, mientras que la apertura se define como primero la aplicación de una dilatación y después una erosión.

En procesamiento de imágenes estos operadores se utilizan comunmente para mejorar la *calidad"* de las formas de los objetos binarios, donde la *calidad"* puede ser un concepto muy subjetivo y que dependerá de la aplicación en cuestión. Una de las aplicaciones de la apertura es por ejemplo para eliminar pequeñas porosidades en los objetos: al dilatar estas porosidades se cierran y una erosión restiruye de manera aproximada el tamaño del objeto. Por su parte la cerradura se utiliza muy comunmente para eliminar objetos muy pequeños los cuales se pueden presentar por ruido: la erosión los elimina y la dilatación regenera de manera aproximada el tamaño de los objetos grandes.

5.2.4 Conteo de objetos sin huecos, basado en configuraciones elementarias

Cuando es el caso de una aplicación en la cual se desean contar los componentes conectados que hay una imagen binaria, y se sabe de antemano que los objetos en la imagen no tienen huecos, es posible utilizar una técnica muy eficiente la cual no encuentra las etiquetas de todos los píxeles, sino únicamente cuenta el número de objetos en la imagen [15]. Por supuesto estas restricciones en el desempeño del procedimeinto lo hacen muy eficiente en tiempo.

Figure 11: Configuraciones elementales para el conteo de objetos.

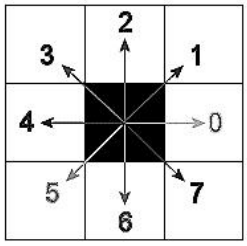


Figure 12: Código de Freeman.

El algoritmo trabaja de la siguiente manera. Se definen 3 configuraciones elementales en una rejilla formada por 4 píxeles vecinos, las cuales son mostradas en la figura 11 y son denominadas E_1, E_3 y E_6. Los puntos negros indican puntos que pertenecen al objeto y los puntos blancos indican puntos que pertenece al fondo. A continuación se recorre la imagen completamente y utilizando un estructura de 4 puntos se cuenta cuantas veces ocurre en la imagen la estructura E_i. Denotamos por N_i el número de veces que ocurre la confuguración elementraria E_i. El numero de objetos en la imagen esta dado por :

$$N_{obj} = N_1 + N_3 - N_6 \tag{43}$$

5.3 Representación del contorno por medio del código de Cadena (Chain Code)

El código de cadena es una estructura de datos que se usa para representar el borde de un objeto. En vez de almacenar las posiciones de todos los puntos del borde, se selecciona un punto de inicio en el borden (del cual si se almacena su coordenada) por un rastreo linea por linea. Entonces se sigue el contorno en la dirección de las manecillas del reloj. En un esquema de vecindad de 8 vecinos hay 8 posibles direcciones y en uno de 4 vecinos existen 4 direcciones posibles para tomar, las cuales pueden ser almacenadas codificadas con 3 o con 2 bits segun sea el caso. La convención utilizada es el codigo de Freeman, el cual se muestra en la figura 12. Por ejemplo un cuadrado de 2×2 píxeles es almacenado como $\{x_o, y_0, 0, 6, 4, 2\}$.

Las ventajas que presenta la utilización del código de Freeman son las siguientes:

1. Es una representación compacta de un objeto binario. Por ejemplo veamos el caso de un disco de diametro D: si el cuadro que limita un objeto tiene D píxeles de ancho, se necesita alrededor de D^2 píxeles y por lo tanto bits. Si por el contrario se utiliza una vecindad de 8, el disco tiene alrededor de πD puntos de frontera, el código de cadena, puede ser almacenado entonces usando $3\pi D$, lo cual es una representación mas compacta para objetos con diámetro es mas grande de 10.

2. Esta representación es invariante a traslaciones, lo cual puede ser una representación útil para comparar objetos. Mas adelante se verá como es util en el cálculo del Area.

3. El código de cadena es una representación completa de un objeto o de una curva, por lo que, en teoría se puede calcular cualquier característica de forma a partir de él.

Las limitantes son que, por supuesto tenemos una representación de la forma de manera discreta, es decir con la representación máxima de la rejilla de la imagen. Si el objeto no es conectado o tiene huecos, es necesario mas de un código de cadena para representarlo.

5.4 Parámetros de forma Simples

Cuando se ha binarizado la imagen, separando los objetos del fondo (o bien se tiene una imagen en escala de gises y un umbral con el cual se separa estas dos clases), y se ha hecho un análisis de componentes conectados, es posible conocer ciertas caracterśticas de los objetos resultantes. Mediante el cálculo de dichas características es posible reconocer un tipo de objetos sobre otros posibles, por ejemplo reconoces objetos de forma circular separándolos de objetos con forma de rectangular. A continuación se muestran algunas características que se pueden calcular sobre los objetos.

5.4.1 Área

Una de las características mas simples e intuitivas de un objeto es el área, que en nuestras imágenes digitales no es otra cosa que el número de píxeles que contiene el objeto.

Por supuesto se puede econtrar el rectángulo que encierra completamente al objeto (bounding box) y contar iterativamente los píxeles que pertenecen al objeto, con lo cual se obtiene el area. Pero una manera mas eficiente de hacerlo es utilizando el código de cadena descrito en la sección 5.3.

Si una imagen esta representada por un contorno codificado en código de cadena, el c'alculo del área se realiza por un algoritmo que trabaja en forma similar a la integración numérica. Este algoritmo asume una lńea base horizontal, trazada en una posición vertical arbitraria de la imagen. Se debe definir una distancia entre la lńea base y el punto inicial del contorno p_i, denotada por B. La integración debe comenzar desde p_i, siguiendo luego el contorno. El área del objeto debe incrementarse o decrementarse, de acuerdo a la dirección almacenada en el código. Por ejemplo, si nos movemos a la derecha sobre el borde (código de cadena 0), el área se incrementa en un factor de B. Si nos movemos hacia arriba, (código de cadena 1) el area nuevamente se incrementa por B, pero B debe de ser tambien incrementada. Las reglas completas del algoritmo se muestran en la tabla 3

5.4.2 Perímetro

Es un parámetro geométrico que al igual que el área, puede ser calculado a partir del cd́igo de cadena. Para realizar este cĺculo, es necesario contar la longitud del código, y tomar en consideración que los pasos en direcciones diagonales deben ser multiplicados por un factor igual a raíz cuadrada de dos. El perímetro p de un contorno codificado en cd́igo de cadena se expresa como:

$$p = n_e + \sqrt{2} n_o \tag{44}$$

donde n_e representa el nḿero pasos con identificador par del código y n_o el número pasos impares del código.

Codigo de contorno	Incremento en el Area	Incremento en B
0	$+B$	0
1	$+B$	1
2	0	1
3	$-B$	1
4	$-B$	0
5	$-B$	-1
6	0	-1
7	$+B$	-1

Table 3: Cálculo del Area de un objeto en base a su código de cadena.

5.4.3 Circularidad

La circularidad es una medida de que tan parecido es un objeto a un círculo. Puede ser expresada como la varianza de la distancia de los píxeles del borde al centroide del objeto. Si un objeto es perfectamente circular, la varianza debe de ser cero.

El algoritmo es sencillo:

1. Se calcula el centroide del objeto.
2. Para cada punto del borde se calcula la distancia euclidiana de este punto al centroide y se almacena en un vector.
3. Se calcula la varianza de este vector de medidas y se toma esta como valor de circularidad.

5.4.4 Compacidad

Es uno de los parámetros geométricos utilizados en la comparación de objetos que son observados desde diferentes distancias, ya que el mismo no depende del tamaño del objeto. La compacidad es un número adimensional definido como:

$$F = \frac{4\pi A}{p^2} \tag{45}$$

El valor de compacidad será mínimo para objetos circulares, pero esta medida es sensible al ruido. Objetos con bordes ruidosos tendrán un perímetro muy largo en comparación con objetos que tienen bordes suaves, por lo tanto se obtendrán valores distintos de compacidad no obstante la circularidad de ambos objetos.

5.4.5 Rectángulo de mínima area que contiene al objeto (Bounding box)

Otro parámetro muy simple para describir crudamente el área de un objeto es el rectángulo de mínima area que contiene el área. Por supuesto este rectángulo esta definido por las coordenadas mínimas y máximas del objeto tanto verticales como horizontales. La desventaja obvia de un descriptor tan sencillo, es que es no es invariante a rotaciones.

5.4.6 Descriptores de forma basados en Momentos

A continuación se definiran los *momentos* para imágenes binarias y en escala de grises, y entonces e mostrará como extraer parámetros de forma útiles para nuestros propósitos.

Se define los momentos de una función bi-dimensional de valores de gris $g(\mathbf{x})$ de un objeto como:

$$m_{p,q} = \int (x_1 - \langle x_1 \rangle)^p (x_2 - \langle x_2 \rangle)^q g(\mathbf{x}) d^2x \tag{46}$$

donde

$$\langle x_i \rangle = \frac{\int x_i g(\mathbf{x}) d^2x}{\int g(\mathbf{x}) d^2x} \tag{47}$$

Donde la integración incluye el area del objeto. Lo anterior se puede entender dado que el vector $\langle \mathbf{x} \rangle = (\langle x_1 \rangle, \langle x_1 \rangle)$ es llamdo el centro de masa del objeto. Si tomamos $g(\mathbf{x})$ como la densidad $p(\mathbf{x})$, por lo tanto el momento de orden cero $m_{0,0}$ nos indica la masa total del objeto. En el caso de nuestras imágenes discretas binarias el cálculo de los momentos se reduce a :

$$m_{p,q} = \sum_{x_1, x_2} (x_1 - \langle x_1 \rangle)^p (x_2 - \langle x_2 \rangle)^q \tag{48}$$

Si se está interesado únicamente en los parámetros de forma de un objeto podemos utilizar los momentos basados en una imagen binaria. Sin embargo, los monentos basados en escala de grises reflejan la distribución de características dentro del objeto.

Es importante utilizar momentos normalizados, los cuales sean invariantes a escala. Si un objeto es escalado por un factor α, $g'(\mathbf{x}) = g(\mathbf{x}/\alpha)$, el momento es escalado por

$$m'_{p,q} = \alpha^{p+q+2} m_{p,q} \tag{49}$$

Por lo tanto, podemos normalizar los momentos con el momento de orden cero, $m_{0,0}$ para obtener momentos invariantes a escala:

$$\overline{m}_{p,q} = \frac{m_{p,q}}{m_{0,0}^{(p+q+2)/2}} \tag{50}$$

Ya que el momento de orden cero de objetos binarios nos indica el área del mismo, los momentos normalizados estan escalados por área, por ejemplo los momentos de de segundo orden ($p + q = 2$) estan escalados por el cuadrado del area, etc..

Como los momentos de orden 1 son zero por definición, el análisis de forma comienza apartir de los momentos de segundo orden. $m_{2,0}$, $m_{0,2}$ y $m_{1,1}$ contienen términos en los cuales la función de valores de gris son multiplicados por el cuadrado de la distancia del centro de masa. Esos términos son los terminos del tensor de inercia para la rotación del objeto alrededor del centro de masa:

$$J = \begin{pmatrix} m_{2,0} & -m_{1,1} \\ -m_{1,1} & m_{0,2} \end{pmatrix} \tag{51}$$

La orientación del objeto es definido como el ángulo entre el eje x y el eje en el cual el objeto puede ser rotado con la mínima inercia. Este es el eigen-vector con el mínimo eigen-valor, en la dirección donde el objeto es mas estirado, como se muestra en la figura 13. Este ángulo se puede calcular como:

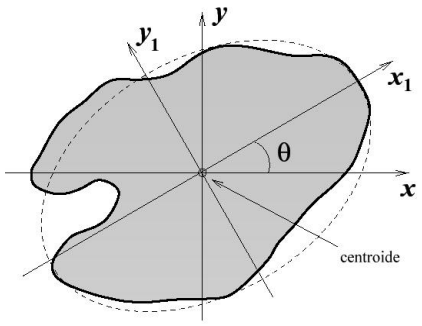


Figure 13: Ejes del tensor de inercia basados en los momentos de segundo orden de un objeto.

$$\Theta = \frac{1}{2} atan \frac{2m_{1,1}}{m_{2,0} - m_{2,0}} \tag{52}$$

Por otro lado la excentricidad del elipsoide asociado (ver figura 13) es calculada como una medida de coherencia, la cual toma valores entre [0,1] (0 para círculos y 1 para formas de palo):

$$\epsilon = \frac{(m_{2,0} - m_{0,2})^2 + 4m_{1,1}^2}{(m_{2,0} + m_{0,2})^2} \tag{53}$$

5.4.7 Dispersión y elongación

Basándose en los momentos calculados en la sección 5.4.6 y ne los momentos crudos (raw moments) es posible calcular otros descriptores de forma como a continuación se ilustra.

En las imágenes discretas los momentos crudos de orden $n = p + q$ se definen como:

$$\mu_{p,q} = \sum_{x_1, x_2} (x_1)^p (x_2)^q \tag{54}$$

Usando lo anterior, la dispersión esta definida por:

$$S = \frac{\mu_{2,0} - \mu_{0,2}}{m_{0,0}^2} \tag{55}$$

y la elongación esta definida por:

$$E = \frac{2m_{0,0}^2[4\mu_{1,1}^2 + (\mu_{2,0} - \mu_{0,2})^2]^{1/2}}{\mu_{2,0} + \mu_{0,2}} \tag{56}$$

La dispersión mide que tan desigualmente esta distribuida alrededor del centroide y la elongación mide el grado de concentración de la masa a lo largo del eje principal.

5.4.8 Radio mínimo y máximo del centro al borde

Otro descriptor de forma es el cálculo del centro al borde del objeto. Proceso que se puede llevar a cabo mediante el recorrido secuencial de los puntos del borde y midiendo la distancia al centroide. La figura 14 muestra como esta medida presenta variaciones para distintas formas de objetos. Así mismo es posible

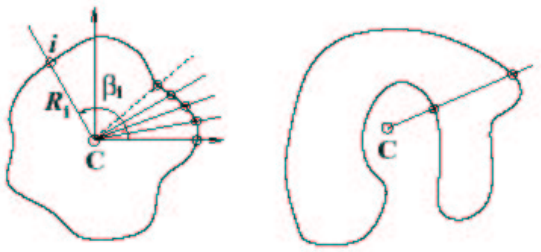


Figure 14: La distancia del centro del objeto al borde como un descriptos de forma.

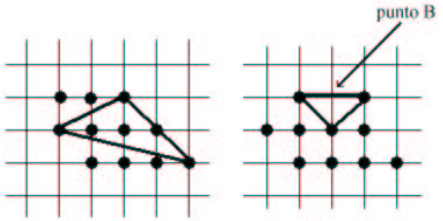


Figure 15: Izquierda: Un objeto convexo, derecha un objeto cóncavo .

facilmente imaginar objetos muy distintos que sin embargo tiene las mismas distancias mínimas y máximas. Como todos los descriptores de forma, es la aplicación específica la que indicara si es viable o no utilizarlo.

5.4.9 Determinación de la convexidad del objeto

Otro parámetro del objeto es si este es cóncavo o convexo. En una imagen discreta es posible definir un objeto convexo si para cualesquiera 3 puntos del objeto, todos los puntos (de coordenadas enteras) contenidos en el triángulo que forman pertenecen al objeto. La figura 15 muestra tanto un objeto convexo como un objeto cóncavo. Podemos detectar que el objeto de la derecha es concavo ya que el triángulo definido por 3 puntos contiene un punto (punto B) que no pertenece al objeto.

La manera de saber si todos los puntos del objeto estan contenidos en todos los posibles triángulos es mediante la selección de todas las triadas posibles de puntos (p_1, p_2, p_3) no colineales del objeto. Luego entonces las coordenadas de cualquier punto en el plano puden ser representadas como:

$$x = x_1 + \lambda(x_2 - x_1) + \mu(x_3 - x_1) \tag{57}$$

$$y = y_1 + \lambda(y_2 - y_1) + \mu(y_3 - y_1) \tag{58}$$

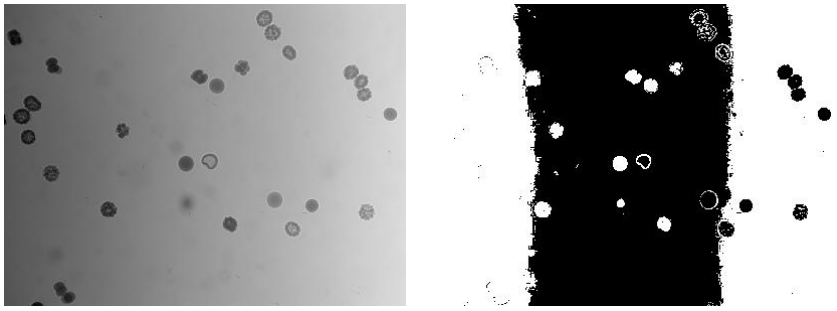
con

$$\lambda = \frac{(x - x_1)(y_3 - y_1) - (y - y_1)(x_3 - x_1)}{(x_2 - x_1)(y_3 - y_1) - (y_2 - y_1)(x_3 - x_1)}, \tag{59}$$

$$\mu = \frac{(x - x_1)(y_2 - y_1) - (y - y_1)(x_2 - x_1)}{(x_3 - x_1)(y_2 - y_1) - (y_3 - y_1)(x_2 - x_1)}. \tag{60}$$

$$\tag{61}$$

Con los cálculos anteriores, es posible saber si el punto de coordenadas (x, y) está dentro del triángulo, ya que si es así se debe de cumplir que $\lambda > 0$, $\mu > 0$ y $\lambda + \mu \leq 1$. El cálculo descrito anterioremente es



a) b)

Figure 16: a) Imagen con iluminación inhomogenéa. b) Resultado de aplicar un umbralizado.

computacionalmente costoso, ya que para cada objeto con n puntos esta validación se debe de hacer $\begin{pmatrix} n \\ 3 \end{pmatrix}$ veces.

6 Experimentos Ilustrativos

En esta sección se seleccionaron algunos de los métodos expuestos en las secciones anteriores, así como imágenes que con características diferentes (que presentan algunos de los problemas vistos en la sección 3). Lo anterior con el objetivo de mostrar algunos experimentos que den una idea de los problemas prácticos que surgen cuando se desarrollan tareas de búsqueda, detección y conteo de objetos.

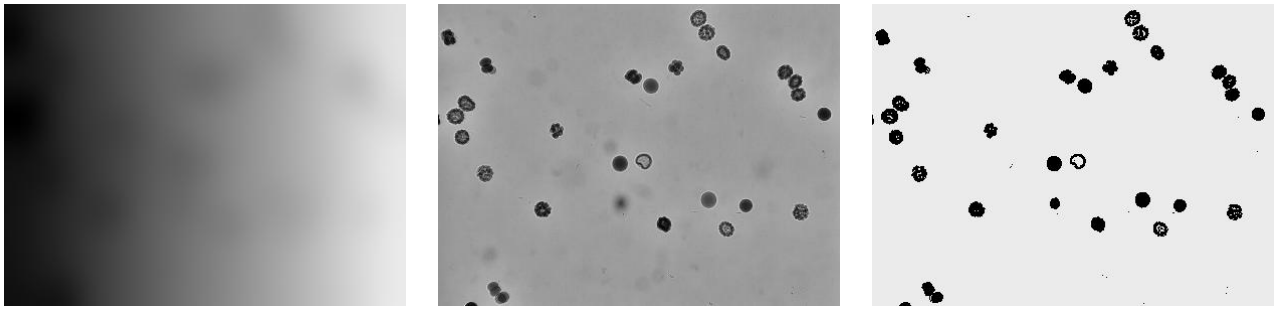
6.1 Segmentación basada en píxeles

En esta sección ilustramos el tipo de segmentación basada en umbralizado. Se ilustra ademas los casos con problemas de shading y una solución a los mismos.

En la figura 16a se puede apreciar una imagen (200×200 píxeles) en la cual se desea segmentar los objetos pequeños obscuros. En este caso la imagen tiene el problema de la iluminación (generado sintéticamente), de tal forma que, como se muestra en la figura 16a, la aplicación de un umbral (ver sección 4.1) fracasa ya que las intensidades de los píxeles no indican si pertenecen o no al fondo.

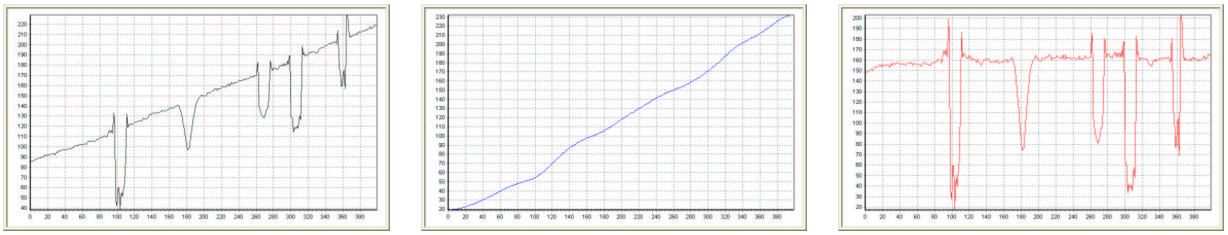
Ante estos problemas, es necesario entonces hacer una corrección de la iluminación (ver sección 4.1.1). El proceso que se llevó a cabo en este caso fué uno muy sencillo: se toma la imagen original y se hace pasar por un filtro pasabajas. El resultado de dicho proceso se ve en la figura 17a. Esta imagen es substraida de la imagen original dándo como resultado la imagen mostrada en 17b. Por último, esta nueva imagen corregida es segmentada aplicando un método de umbralizado y los resultados son mostrados en la figura 17c.

Este proceso de corrección se pude apreciar en las gráficas de un renglón de la imagen mostradas en la figura 18. En esta figura se puede apreciar a) el renglón original, b) el renglón filtrado y c) el renglón corregido. Como es lógico suponer, este método es muy barato computacionalmente hablando, tanto en la implementación como en la ejecución, ya que el filtrado puede hacerse simplemente como un proceso de promediación de cada pixel con sus vecinos y la aplicación de un umbral es una tarea muy rápida.



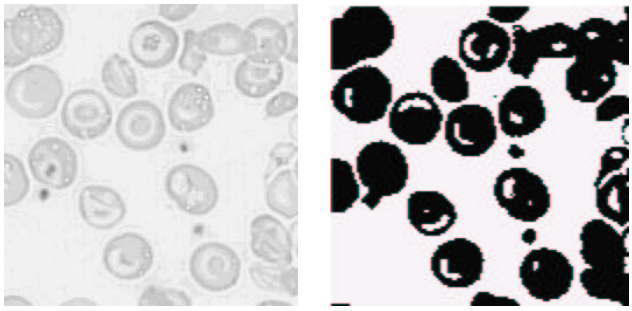
a) b) c)

Figure 17: a) Resultado de la aplicación de un filtro pasabajas a la imagen con iluminación inhomogénea b) Corrección al restarle la imagen filtrada. C) Segmentación por medio de umbral sobre la imagen corregida



a) b) c)

Figure 18: Graficas de un renglón en la imagen a) Con problema de Shading, b) Resultado de la aplicación de un filtro pasabajas c) imagen corregida.



a) b)

Figure 19: a) Imagen original. b) Resultado de 5 iteraciones de algoritmo EM

6.2 Segmentación basada en regiones

Para este experimento se utilizó una imagen de células (128 × 128 píxeles), la cual se puede ver en la figura 19a, y se utilizó un algoritmo EM (ver sección 4.2.6). El resultado de 5 iteraciones es mostrado en la figura 19b. Los parámetros iniciales de las clases fueron $\mu_o^i = 0.1$, $\mu_f^i = 0.95$, $\sigma_o^i = 0.1$ y $\sigma_f^i = 0.1$ (el rango dinámico de la imagen fué normalizado de 0 a 1). Los parámetros encontrados por el método fueron $\hat{\mu}_o = 0.394$, $\hat{\mu}_f = 0.726$, $\hat{\sigma}_o = 0.133$ y $\hat{\sigma}_f = 0.055$.

Como podemos ver, la desviación estandar del los objetos es más grande que la del fondo, lo cual indica que el modelo es consistente con la realidad, ya que hay mas escalas de gris en los objetos que en el fondo. Podemos ver también que el algoritmo es robusto a la inicialización de los parámetros, ya que los estimadores iniciales son significativamente distintos a los encontrados.

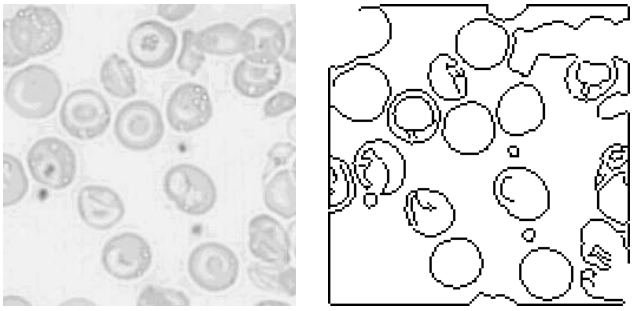
Nótese que en este caso los objetos tienen huecos, los cuales, dependiendo de la aplicación deben de ser removidos para obtener objetos compactos.

La aplicación de un algoritmo de este tipo, conlleva un proceso de programación elaborado. La ejecución del mismo toma un tiempo considerable (poco mas de 1 minuto en una computadora Pentium 1.6 GHz). Sin embargo, los resultados por lo general son de alta calidad y tiene el valor agregado de encontrar los parámetros del modelo. Esto último puede agilizar el procesamiento de imagenes futuras. Si se asume que se procesarán varias imágenes con condiciones de captura similares, en el procesamiento de estas los parámetros obtenidos se pueden introducir como parámetros iniciales, logrando que con 1 o 2 iteraciones sea suficiente.

6.3 Segmentación basada en Bordes

Para este tipo de experimentos se implementó el detector de bordes de Canny (ver seccción 4.3.1). La imagen usada para este experimento se muestra en la figura 20a (la cuál es la misma usada en el experimento anterior). Los bordes obtenidos con los parámetros $\sigma = 1.0$, $T_H = 0.75$ y $T_L = 0.15$ son mostrados en la figura 20b.

Como podemos ver, dependiendo de los parámetros, algunos bordes son cerrados u otros no, lo cual representa un problema si se desea "rellenar" los objetos. Es quizá viable (dependiendo de la aplicación) detectar los bordes no cerrados por medio de los extremos (aquellos píxeles de bordes que no tienen mas que un vecino) y eliminarlos. Este problema de bordes abiertos presenta una desventaja con respecto a la aplicación del algoritmo EM, sin embargo este método es mas fácil de implementar y el tiempo de ejecución



a) b)

Figure 20: a) Imagen original. b) Bordes encontrados por el algoritmo de Canny.

es considerablemente menor (1 segundo en la misma PC).

6.4 Segmentación basada Textura

Para los experimentos de esta sección se escogieron algunas imágenes con textura, las cuales, es casi imposible segmentarlas en base a su valor de gris. Los métodos probados fueron un banco de filtros de Gabor (ver sección 4.4.4) y los kernels de Laws (ver sección 4.4.2).

La figura 21 muestra los resultados de esta segmentación, en ella se muestra por renglones los experimentos, la primera columna contiene las imágenes originales (200 $\times$ 200 píxeles en promedio) , la segunda columna tiene los resultados de la segmentación con el banco de filtros de Gabor y la tercera columna muestra los resultados de la aplicación de los kerneles de Laws.

A continuación se indican los parámetros utilizados. En todos los caso se colocó un banco de 40 filtros de Gabor equidistantes en el espacio de la frecuencia y con la misma anchura $\sigma = 3.0$. La segmentación que se presenta corresponde a una agrupación usando el algoritmo de K-Medias, con k = 2 (ver sección 4.5.1) tomando como entrada de características las respuestas de los filtros para cada pixel.

En el caso de los kernels de Laws, el unico parámetro que es el tamaño de la vecindad del filtrado, el cual se fijó de esta manera : 17, 25 y 17, para el primer, segundo y tercer experimento respectivamente. La agrupación se hizo igualmente usando el algoritmo de K-Medias con el parámetro k = 2.

Como podemos ver en los resultados, la segmentación en textura es un proceso complicado, dependiendo de los parámetros escogidos los objetos se pueden obtener erosionados o bien dilatados. El caso extremo se ve en el panel 21b en donde los filtros de gabor no pudieron segmentar correctamente el objeto.

El proceso de Gabor es significativamente mas caro que el utilizar las energias de Laws. La ejecución de los filtros de Gabor involucra convoluciones y/o transformadas de Fourier directas e inversas, lo cual hace el proceso lento (aproximadamente 50 segundos en la PC descrita anteriormente, implementación en lenguaje C). Por su parte las energías de Laws son de fácil implementación y su ejecución es rápida (6 segundos, misma computadora, implementación en Matlab).

6.5 Descriptores de forma

Finalmente se muestra un experimento en el cual se desea diferenciar un tipo de objetos de otros.

La figura 22 muestra el proceso completo desde la imagen original (panel a), la aplicación de un umbral

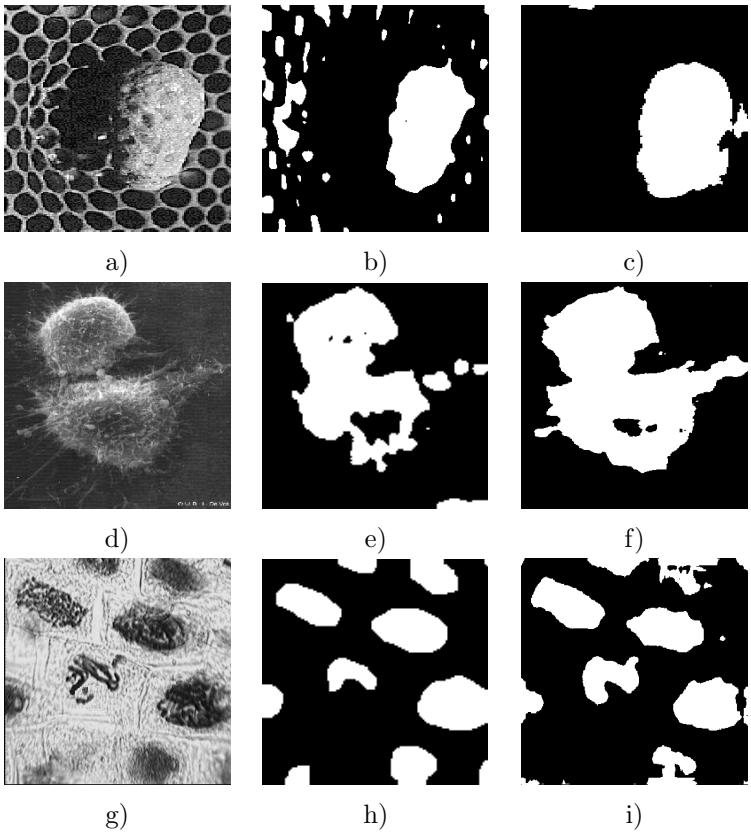


Figure 21: Segmentación de textura, experimentos por renglones. La primera columna tiene las imágenes originales, la segunda columna tiene los resultados por medio de un banco de filtros de gabor y la tercera columna los resultados de la aplicación de los kernels de laws.

(panel b), el rellenado de los huecos (panel c), la aplicación de una cerradura para eliminar los objetos muy pequeños (panel d) y finalmente el proceso de etiquetado (ver sección 5.1).

Una vez que se tienen estos objetos en una imagen binaria, ahora es necesario poder determinar características de cada uno de ellos para poder identificarlos en dado caso de que haya mas de un tipo de objeto, o tambíen si se desea detectar traslapes de objetos, las cuales dan como resultados componentes que no tienen formas elípticas como se puede ver en la figura 22d.

Para mostrar la utilidad y (problemas que se presentan) cuando se escoge algun descrptor de forma, se seleccionaron 4 de los objetos segmentados para encontrar sus coeficiente de compacidad (ver sección 5.4.4). La figura 23 muestra los 4 objetos seleccionados y la tabla 4 mustra los coeficientes de compacidad calculados para cada uno de ellos.

Es importante hacer notar que salvo el objeto mas circular [panel 23b] el cual tiene un valor de compacidad significativamente mas grande que los demas y el objeto que tiene un gran hueco [panel 23c], el cual tiene un valor de compacidad significativamente inferior, nos sería muy difícil poder separar el caso de los objetos con forma de estrella [panel 23a] de los casos donde 2 objetos elípticos se intersectaron [panel 23d], ya que sus coeficientes no son muy diferentes. Sin embargo, si consideramos que el tamaño medio de los objetos es similar, y nos fijamos en los perímetros o el área, el caso de los objetos traslapados es quizá identificable

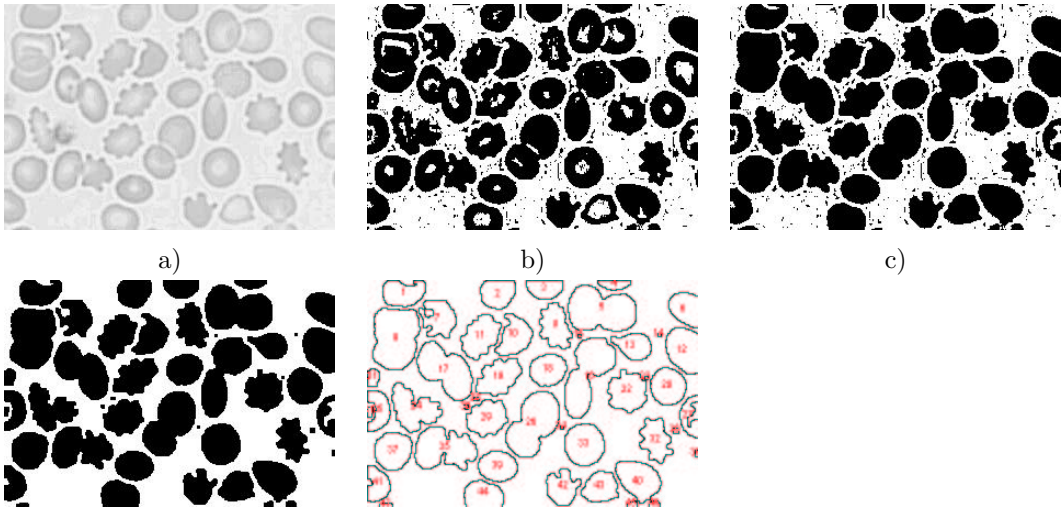


Figure 22: Proceso completo, desde umbralizado, restauración y análisis de componentes conectados.

Figure 23: Objetos encontrado en aplicaciones prácticas de formas variadas.

por esta medida, la cual podemos apreciar que es significativamente mayor. El caso es que, nuevamente, dependiendo de la aplicación, es posible que unos descriptores de forma nos ayuden o no, y que algunos descriptores sencillos trabajen mejor que otros mas complicados.

7 Conclusiones

El número de estrategias computacionales que se han desarrollado para la llevar a cabo la búsqueda detección y conteo de objetos en imágenes digitales es sumamente extenso. Sin embargo, no existen esquemas de solución generales para todos los casos posibles. Existen métodos que por definición son especializados para ciertos casos y métodos que aunque parecieran de aplicación general, siempre es posible encontrar casos en los cuales no van a entregar los resultados deseados (ya que, debido a la creciente demanda de aplicaciones automáticas o semi-automáticas de vision computacional, los casos de aplicación son también muchos, son complicados por naturaleza y de índole muy variada).

Por lo anterior, es necesario tener en mente, que cuando se desea solucionar un problema práctico, se necesita analizar el problema como un paso primario (y fundamental), lo cual nos conducirá a buscar un método de solución que se adapte adecuadamente a las características del problema en cuestión. Este trabajo, da un panorama ámplio de los diferentes problemas y las diferentes esquemas de solución que se han

Objeto	Area	Perímetro	Compacidad
a)	903.5	135.98	0.614027677
b)	727	104.08	0.843353609
C)	720.5	160.46	0.351649718
d)	1568.5	178.26	0.620278292

Table 4: Medida de compacidad para los objetos de la figura 23

propuesto, de tal manera, que sirva como un punto de partida en la búsqueda de la solución mas adecuada (o en el mejor de los casos indique una manera de resolver el problema).

En la búsqueda de soluciones, un adecuado análisis del problema nos puede llevar a soluciones sencillas bien adaptadas al problema, las cuales pueden dar mejores resultados que la aplicacion de métodos un tanto complicados de caracter genérico.

En la sección de resultados se han presentado los productos obtenidos con algunos de los métodos explicados en las secciones anteriores. Esto con el objetivo de conocer o *permearse* en el problema y de dejar un registro de las posibles soluciones obtenidas a algunos casos que sirven de ejemplo.

References

[1] Harry Wechsler "Computational Vision", Academic Press,.

[2] K. Laws. Rapid texture identification. In SPIE Vol. 238 Image Processing for Missile Guidance, pages 376-380, 1980.

[3] Bernard Jahne. Digital Image Processing. Concepts, Algorithms and scientific applications. Springer-Verlag

[4] T. Hastie, R. Tibshirani, J. Friedman "The Elements of Statistical Learning: Data Mining, Inference, and Prediction" Springer-Verlag, 2001

[5] S.Z. Li, "Markov Random Field Modeling in Image Analysis," Springer Verlag, Tokyo, 2001.

[6] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distribution and the Bayesian restoration of images," IEEE Transactions on Pattern Analysis and Machine Intelligence, 6, 6, 721-741, 1984.

[7] Preparata F. P. and M. I. Shamos. *Computational Geometry.* Spreinger Verlag. New York, 1985

[8] Tuceryan, M. and Jain, A.K. Texture Analysis. HPRCV , 1998.

[9] C. H. Chen, L. F. Pau and P. S. P. Wang . HANDBOOK OF PATTERN RECOGNITION AND COMPUTER VISION. World Scientific Publishing Co. 1998.

[10] P. J. Burt, T. H. Hong, A. Rosenfeld. Segmentation and Estimation of Image Region Properties Through Cooperative Hierarchical Computation. IEEE Tran. On SMC, Vol. 11, N.12, 1981, pp. 802-809.

[11] J. Canny. A Computational Approach to Edge Detection, IEEE Trans. on Pattern Analysis and Machine Intelligence, 8(6), pp. 679-698 (1986).

[12] N. Milstein Image Segmentation by Adaptive Thresholding, Technion Israel Institute of Technology, The Faculty for Computer Sciences, 1998.

[13] P. Kruizinga, N. Petkov and S.E. Grigorescu. Comparison of texture features based on Gabor filters. Proceedings of the 10th International Conference on Image Analysis and Processing, Venice, Italy, September 27-29, 1999, pp.142-147.

[14] Fu Chang, Chun-Jen Chen: A Component-Labeling Algorithm Using Contour Tracing Technique. ICDAR 2003: 741-745

[15] K. Voss: Discrete Images, Objects ans Functions in Z^n. Springer Verlag . 1993.