SEGMENTACIÓN DE IMÁGENES POR MEDIO DE SPECTRAL CLUSTERING

JOHN PEÑA JURADO

Trabajo de grado

Director

Ing. Elena Muñoz España

UNIVERSIDAD DEL CAUCA

FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES

DEPARTAMENTO DE ELECTRÓNICA, INSTRUMENTACIÓN Y CONTROL

LINEA DE INVESTIGACIÓN EN ROBÓTICA Y VISIÓN

POPAYÁN, FEBRERO DE 2004

TABLA DE CONTENIDO

1 INTRODUCCIÓN

Segmentación es un término ampliamente utilizado para definir varios procesos efectuados sobre una imagen. En términos generales involucra dividir una imagen en partes que guarden coherencia con respecto a intensidad, color, textura, contornos o formas predeterminadas. Actualmente no existe una teoría general de segmentación de imágenes a pesar de que el tema ha sido abordado desde muy variadas perspectivas.

Últimamente se le ha dado importancia a enfoques congruentes con la psicología de la Gestalt[1] la cual reveló la importancia de la organización y el agrupamiento perceptivos en la visión. Dichos enfoques no analizan las unidades que conforman una imagen, sino que toman en cuenta configuraciones globales, la formulación de *agrupamiento* de la teoría espectral de grafos es coherente con lo anterior y ha permitido el surgimiento de una técnica llamada *spectral clustering*, la cual será tratada en este documento.

Cuando se quiere segmentar una imagen de forma automática y no se posee información previa acerca de esta, la decisión acerca del número óptimo de regiones en que se debe dividir es un problema difícil que aún no se ha solucionado de forma definitiva, sin embargo, cuando se posee información sobre el tipo y nivel de ruido de la imagen se pueden aplicar ciertas técnicas que son de vital importancia para inicializar algoritmos de segmentación. En el presente documento se implementa una técnica propuesta por Vidal [23] conocida como *segmentación polinomial* (*polysegment*) que permite inicializar algoritmos de segmentación y si se desea, segmentar la imagen desde una perspectiva diferente a la de *spectral clustering*.

[1] Escuela de psicología que se dedicó principalmente al estudio de la percepción. Los psicólogos gestaltistas descubrieron que la percepción es influida por el contexto y la configuración de los elementos percibidos; las partes derivan de su naturaleza y su sentido global, y no pueden ser disociados del conjunto, ya que fuera de él pierden todo su significado. Los experimentos de los partidarios de esta teoría muestran que la percepción de la forma no depende de la percepción de los elementos individuales que la constituyen.

Al existir tan diversos tipos de imágenes es muy importante escoger correctamente las características que se van a asociar a cada píxel para realizar la segmentación, tales características están basadas principalmente en intensidad, color, contornos y textura. En el presente documento se integran las técnicas spectral clustering y segmentación polinomial con el desarrollo de una investigación acerca de la extracción de características de una imagen para lograr un proceso matemático que permite una segmentación automática de imágenes. Lo anterior dirigido hacia las necesidades actuales de la Universidad del Cauca en el área de Visión Artificial.

Esta monografía incluye un marco conceptual en el que se explican definiciones básicas que son necesarias para el desarrollo de los capítulos posteriores. En el cuarto capítulo se describe el algoritmo básico de spectral clustering y las diferentes implementaciones de dicho algoritmo que difieren en las características que se extraen de las imágenes, se asumirá que el número de regiones en que se debe dividir la imagen es conocido.

En el quinto capítulo se describe la implementación de la segmentación polinomial que permite determinar el número de regiones de manera automática. En el capítulo seis se enumeran las conclusiones. En el capítulo siete se incluye el código fuente de los principales programas desarrollados.

2 MARCO CONCEPTUAL

2.1 Definiciones básicas de teoría de grafos[2]

Si $[A]^k$ denota el conjunto de todos los subconjuntos con *k* elementos de *A*, un grafo es un par de conjuntos *G = (V, E)* que satisfacen $E \subseteq [V]^2$; de esta manera, los elementos de *E* son subconjuntos de *V* con dos elementos. Los elementos de *V* son los vértices o nodos del grafo *G* y los elementos de *E* son sus bordes o arcos. El conjunto de vértices de un grafo *G* se denota como *V (G)* y su conjunto de bordes como *E (G).*

La forma usual de representar un grafo es dibujar un punto por cada vértice y unir dos de estos puntos por medio de una línea en caso de que los vértices correspondientes formen un borde. A un grafo con conjunto de vértices *V* se le llama un grafo en *V.*

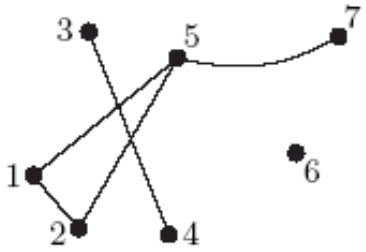

figura 1

El grafo en $V = \{1, 2, 3, 4, 5, 6, 7\}$
con conjunto de bordes
$E = \{ \{1, 2\}, \{1, 5\}, \{2, 5\}, \{3, 4\}, \{5, 7\} \}$

Un vértice *v* incide en un borde *e* si $v \in e$; de esta manera *e* es un borde en *v*. El *orden* de un grafo *G* equivale a su número de vértices. El *grado* de un vértice *v* es el número de bordes en *v*, un vértice de grado cero está *aislado.*

[2] Basado en la información de [**7**], [**14**], [**30**], [**31**], [**32**], [**33**].

Un borde {x, y} usualmente se denota como ***xy*** o ***yx***. Dos vértices ***x***, ***y*** de un grafo ***G*** son *adyacentes* o *vecinos* si ***xy*** es un borde de ***G***. Si todos los vértices de ***G*** son adyacentes por pares entonces ***G*** es un grafo *completo*. $\boldsymbol{K_n}$ denota un grafo completo con *n* vértices.

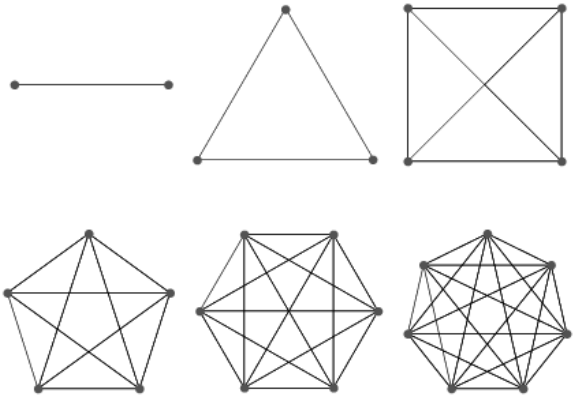

figura 2
Ejemplos de grafos completos:
K_2, K_3, K_4, K_5, K_6.

Un grafo G no vacío es llamado *conectado* si existe un trayecto entre cualquier par de sus vértices. De esta manera para un grafo *completo* siempre es un grafo *conectado* (ver figura 2).

Un grafo *ponderado* es aquel en el cual cada borde posee un valor numérico determinado por alguna función particular aplicada entre sus vértices. Un grafo *ponderado no dirigido G* tiene asociada a él una función $w:VxV \to R$ que satisface: *w(u, v) = w(v, u)* y $w(u,v) \geq 0$ se puede apreciar que si $\{u,v\} \notin E(G)$, entonces *w(u, v) = 0*.

La matriz de adyacencia $A = (a_{ij})_{nxn}$ de un grafo *no ponderado G* esta definida por:

$$a_{ij} = \begin{cases} 1 & si\ v_i v_j \in E(G) \\ 0 & si\ v_i v_j \notin E(G) \end{cases}$$

En la **figura 4** se aprecia un ejemplo.

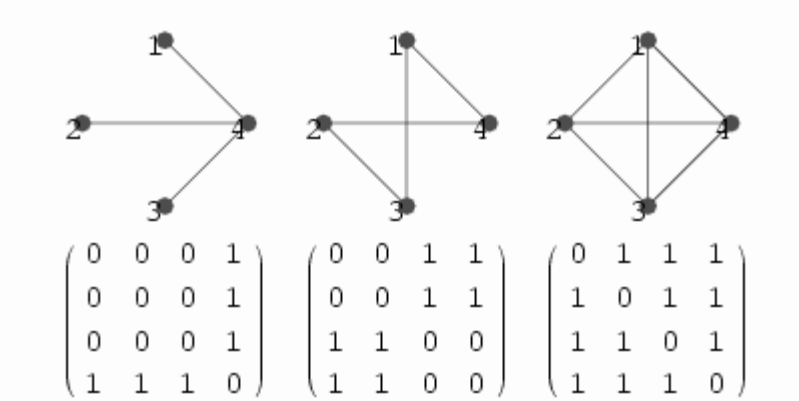

figura 4
Tres grafos y sus respectivas matrices de adyacencia

Dicho de otra manera, la matriz de adyacencia es aquella cuyos elementos se deducen de los vértices del grafo de tal forma que si los vértices (v_i, v_j) son adyacentes el elemento $_{ij}$ de la matriz es igual a 1, de lo contrario es igual a 0. Si se trata de grafos ponderados, si los vértices (v_i, v_j) son adyacentes el elemento $_{ij}$ de la matriz toma el valor del borde correspondiente según la función w aplicada entre sus vértices:

$$a_{ij} = \begin{cases} w(v_i, v_j) & si\ v_i v_j \in E(G) \\ 0 & si\ v_i v_j \notin E(G) \end{cases}$$

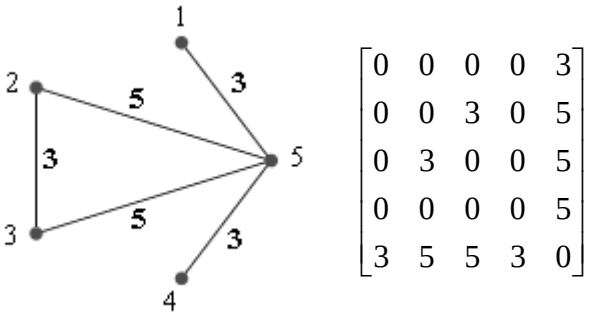

figura 5
Grafo *ponderado* y su respectiva matriz de adyacencia

Una partición de un conjunto es la colección de sus subconjuntos no superpuestos de tal manera que la unión de estos conforma el conjunto original. Una partición de un grafo es una partición tanto de sus vértices *V* como de sus bordes *E* en subconjuntos $\{V_j\}$ y $\{E_j\}$ de tal forma que $G_j = [V_j, E_j]$ también es un grafo. Si $\{G_1, G_2\}$ es una partición del grafo *ponderado* *G*, el grado de disimilaridad entre estas dos partes puede ser calculado como la suma total de los valores de los bordes que han sido removidos para obtener dicha partición, a esto se le llama *corte.* Las figuras 6 y 7 muestran la importancia del concepto de corte en la partición de un grafo. El corte se define como:

$$cut(G_1, G_2) = \sum_{u \in G_1,\, v \in G_2} w(u, v)$$

Las figuras 6 y 7 muestran el corte realizado a un grafo.

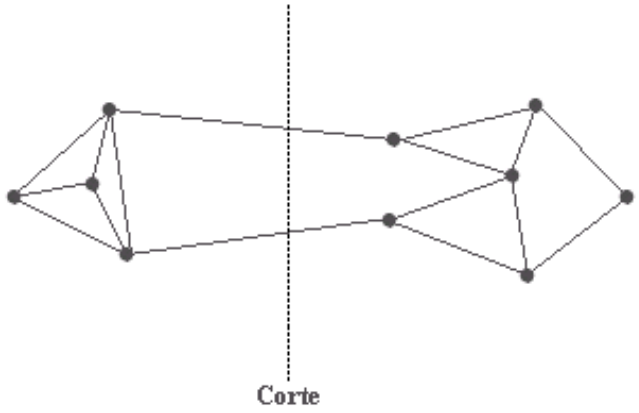

figura 6
E valor del corte es igual a la suma de los valores de los bordes atravesados por la línea punteada, este valor es una medida de disimilaridad entre los grupos resultantes.

figura 7
Una buena partición se obtiene minimizando el valor del corte, esto se logra eliminando los bordes que representan una relación débil entre grupos.

2.2 Matriz de similaridad

La matriz de similaridad es un concepto derivado de la teoría de grafos, representa las relaciones de similaridad entre todos los píxeles de una imagen. Una matriz de similaridad ideal[3] ***S*** asociada a una imagen ***I*** es aquella en la que ***S(i, j)=1*** cuando los píxeles ***i*** y ***j*** pertenecen a una misma región y ***S(i, j) = 0*** si pertenecen a regiones diferentes, ***S*** se puede calcular de la siguiente manera:

$$S(i,j) = \exp\left(\frac{-d^2{}_{ij}}{2\sigma^2}\right) \quad si \quad i \neq j \qquad y \qquad S_{ii} = 0 \qquad \textit{Ecuación (1)}$$

σ es un parámetro que determina que tan rápido decaen las afinidades a medida que aumenta la distancia entre las características y generalmente se calcula experimentalmente. d_{ij} es la distancia[4] entre los vectores que caracterizan a los píxeles **i** y ***j***, de esta manera la anterior ecuación se puede escribir así:

$$S(i,j) = \exp\left(\frac{-Y^2(i,j)}{2\sigma^2}\right) \quad si \quad i \neq j \qquad y \qquad S_{ii} = 0 \qquad \textit{Ecuación (2)}$$

Siendo ***Y*** la matriz de adyacencia de un grafo ponderado cuyos vértices representan los vectores de características asociados a los píxeles de la imagen y cuyos bordes son determinados a partir de la distancia euclidiana entre los respectivos vértices.

[3] En la práctica debido al ruido en las imágenes los valores son aproximados.
[4] En este trabajo se utilizará la distancia euclidiana pero es posible utilizar otro tipo de distancias.

2.3 Matriz laplaciana

La matriz laplaciana se obtiene a partir de la matriz de *similaridad* ***S*** y de la matriz de *grado* ***D*** que consiste en una matriz diagonal determinada por

$$D(i,i) = \sum_{j} S(i, j) \qquad \textit{Ecuación (3)}$$

En otras palabras ***D*** sería una matriz diagonal en la que el elemento ***(i, i)*** sería igual a la suma de los elementos de la fila ***i*** de ***S***. La matriz laplaciana se define como:

$$L = D^{-\frac{1}{2}} * S * D^{-\frac{1}{2}} \qquad \textit{Ecuación (4)}$$

2.4 Teoría espectral de grafos

Los autovalores de una matriz son llamados su espectro. La teoría espectral de grafos es el estudio del espectro de ciertas matrices asociadas a un grafo, tradicionalmente se han estudiado la matriz de adyacencia, la matriz *laplaciana* o sus respectivas formas normalizadas. Una de las principales metas en teoría de grafos es deducir las propiedades y estructura de un grafo a partir de su espectro y sus autovectores.

Como se mencionó anteriormente una buena partición de un grafo asociado a una imagen equivale a una buena partición de los píxeles de dicha imagen en regiones coherentes, esto se logra por medio de la minimización del valor del *corte* entre subconjuntos excluyentes de los vértices del grafo, sin embargo este procedimiento puede conducir a resultados erróneos ya que este criterio favorece la formación de pequeños subconjuntos de vértices aislados. Para resolver este problema Shi y Malik [23] propusieron una modificación en el corte llamada corte normalizado y demostraron que la solución óptima para este corte se encuentra en los autovectores de la matriz laplaciana.

Esto se Aprecia mejor gráficamente. Las imágenes de la figura 8 tienen un número de regiones claramente definidas:

figura 8
Imágenes con 3 y 4 regiones respectivamente.

En las gráficas de los autovectores de las matrices laplacianas asociadas a las anteriores imágenes se aprecia que existe información acerca del número de regiones:

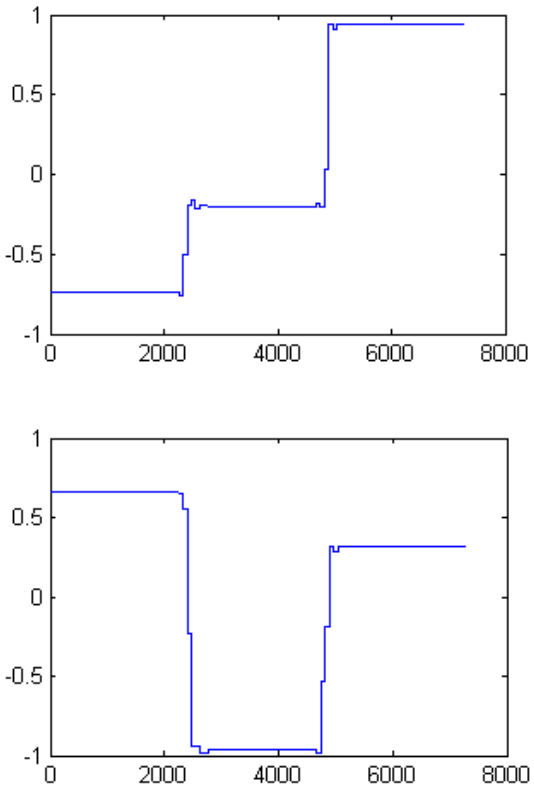

figura 9
Autovectores correspondientes a los 2 mayores autovalores de la matriz laplaciana asociada a la imagen con tres regiones, las gráficas muestran claramente tres grupos, el eje vertical representa el valor del autovector y el eje horizontal el índice del elemento del autovector, ya que la imagen original tiene un tamaño de 97x73 = 7081 píxeles, su matriz laplaciana asociada posee 7081^2 elementos, por lo tanto sus autovectores poseen 7081 elementos, cada uno de ellos relacionado con un píxel de la imagen, los elementos del autovector con valores similares pertenecen a una misma región.

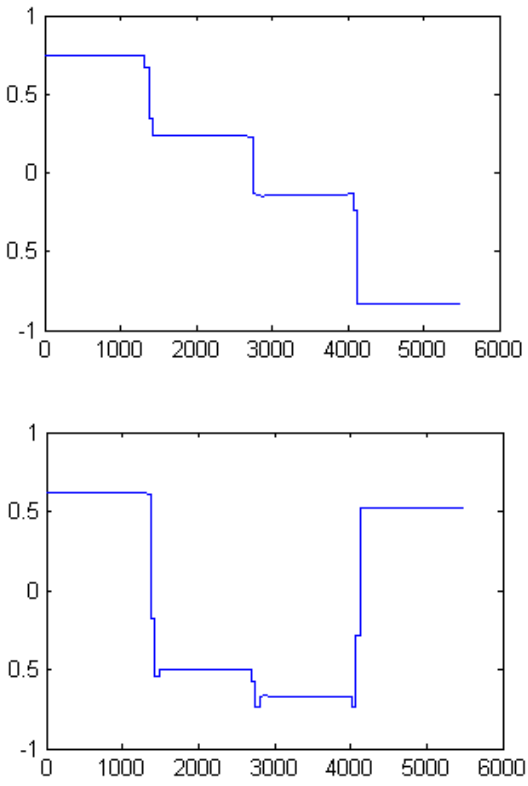

figura 10
Autovectores correspondientes a los 2 mayores autovalores de la matriz laplaciana asociada a la imagen con cuatro regiones, las gráficas muestran claramente cuatro grupos. El eje vertical representa el valor del autovector y el eje horizontal el índice del elemento del autovector cada uno de de los cuales está relacionado con un píxel de la imagen, los elementos del autovector con valores similares pertenecen a una misma región.

A una imagen de ***N*** píxeles corresponde una matriz laplaciana de ***N x N*** elementos, así que cada uno de los autovectores de esta matriz posee **N** elementos, estos están relacionados con los **N** píxeles de la imagen. Idealmente los elementos del autovector que poseen valores iguales corresponden a píxeles que pertenecen a la misma región, en otras palabras para un autovector ***X***, si **X(i) = *X(j)*** entonces los píxeles ***I(i)*** e ***I(j)*** pertenecen a una misma región (asumiendo que la imagen se transforma en un vector columna). En la práctica debido al ruido en las imágenes los valores no son siempre exactamente iguales, para solucionar este problema es común utilizar técnicas como K-means para agrupar los elementos de los autovectores y de esta manera establecer la partición de la imagen.

2.5 K-means

K-means es un método para segmentar datos en un número determinado de grupos mutuamente excluyentes. Para lograr esto cada dato es tratado como un objeto ubicado en un espacio N-dimensional y se agrupan datos de tal forma que las distancias entre datos pertenecientes al mismo grupo sean pequeñas y las distancias entre datos de diferentes grupos sean grandes.

Cada grupo en la partición está definido por sus miembros y por su centro, el centro de cada grupo es el punto en el cual la suma de distancias desde todos sus miembros es minimizada, la distancia utilizada no es necesariamente euclidiana. K-means utiliza un algoritmo iterativo que determina los centros a los que pertenece cada objeto de manera que se obtengan grupos tan compactos y separados como sea posible. Existen varias opciones en cuanto a la escogencia de los valores iniciales para los centros, esto es un aspecto importante para tener en cuenta ya que k-means es una técnica cuyos resultados son muy sensibles a la inicialización.

En el presente trabajo k-means es utilizado para agrupar los elementos de los autovectores de la matriz laplaciana, sin embargo se puede utilizar K-means directamente sobre los vectores que caracterizan la imagen, el problema radica en que K-means no toma en cuenta relaciones globales y puede conducir a particiones erróneas.

2.6 La extensión de Nyström

El método de Nyström es una técnica para encontrar aproximaciones numéricas a problemas de eigenfunciones[5] de la forma

$$\int_a^b W(x,y)\phi(y)dy = \lambda\phi(x) \qquad \textit{Ecuación (4)}$$

Esta ecuación integral puede aproximarse evaluándola en un conjunto de puntos uniformemente espaciados $\xi_1, \xi_2, \ldots \xi_3$ en el intervalo $[a, b]$ y empleado la regla:

$$\frac{(b-a)}{n}\sum_{j=1}^{n} W(x,\xi_j)\hat{\phi}(\xi_j) = \lambda\hat{\phi}(x) \qquad (1) \qquad \textit{Ecuación (5)}$$

En la cual $\hat{\phi}$ es una aproximación de ϕ . Para resolver **(1)** establecemos $x = \xi_i$ obteniendo el siguiente sistema de ecuaciones:

$$\frac{(b-a)}{n}\sum_{j=1}^{n} W(\xi_i,\xi_j)\hat{\phi}(\xi_j) = \lambda\hat{\phi}(\xi_i) \qquad \forall i \in \{1...n\} \qquad \textit{Ecuación (6)}$$

Sin pérdida de generalidad hacemos $[a, b] = [0, 1]$ y estructuramos el sistema para que tome la forma del problema de hallar los autovalores de una matriz

$$A\hat{\phi} = n\hat{\phi}\Lambda \qquad \textit{Ecuación (6)}$$

[5] f es una eigenfunción para L y λ es el autovalor asociado siempre y cuando $Lf = \lambda f$

Donde $\mathbf{A_{ij}} = \mathbf{W(\xi_i, \xi_j)}$ y $\phi = [\phi_1, \phi_2 ... \phi_n]$ son los ***n*** autovectores de ***A*** con los correspondientes autovalores $\boldsymbol{\lambda_1, \lambda_2, \ldots \lambda_n}$. Sustituyendo en **(1)** obtenemos la extensión de Nyström para cada $\hat{\phi}_i$

$$\hat{\phi}_i(x) = \frac{1}{n\lambda_i} \sum_{j=1}^{n} W(x, \xi_j) \hat{\phi}_i(\xi_j) \qquad \textbf{\textit{Ecuación (7)}}$$

Esta expresión nos permite calcular los autovectores de ***A*** utilizando un pequeño porcentaje de sus elementos.

2.7 La extensión de Nyström aplicada al problema de encontrar los autovectores de una matriz de similaridad asociada a una imagen[6]

Ya que la matriz de similaridad de una imagen representa las afinidades entre todos sus píxeles, su tamaño suele ser demasiado grande[7], esto hace que sea difícil computar dicha matriz y complica también la tarea de encontrar los autovectores de la matriz laplaciana asociada. El método de Nyström permite calcular los autovectores de la matriz laplaciana procesando un pequeño porcentaje de los elementos de la matriz de similaridad. Los pasos para implementar el método de Nyström son los siguientes:

1. Convertir la matriz de características asociadas a los píxeles en el vector ***C*** encadenando sus columnas de la siguiente manera asumiendo una imagen de ***m x n*** píxeles:

$$\textit{Matriz de características}\begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mn} \end{bmatrix} \rightarrow \textit{Vector } C \begin{bmatrix} c_{11} \\ c_{21} \\ \vdots \\ c_{m1} \\ c_{12} \\ c_{22} \\ \vdots \\ c_{m2} \\ \vdots \\ \vdots \\ c_{1n} \\ c_{2n} \\ \vdots \\ c_{mn} \end{bmatrix}$$

[6] En el presente trabajo se utiliza la implementación propuesta en [**9**], dicha implementación utiliza conceptos avanzados de álgebra lineal y están fuera del alcance de este trabajo, por lo tanto solo se hará una descripción general del algoritmo.

[7] Para una imagen de 150 x 150 píxeles o sea 22500 píxeles la matriz de similaridad tendrá 22500x 22500 elementos o sea 506250000 elementos.

2. Extraer un pequeño número ***p*** de muestras aleatorias del vector ***C*** [8] y conformar el vector columna ***M***. Con los elementos restantes conformar el vector fila ***R*** que contiene ***(n x m) – p = r*** elementos.

3. Calcular las distancias entre los elementos del vector ***M*** y conformar la matriz ***A*** de la siguiente manera:

$$\begin{bmatrix} A(1,1) & A(1,2) & \cdots & A(1,p) \\ A(2,1) & A(2,2) & \cdots & A(2,p) \\ \vdots & \vdots & \ddots & \vdots \\ A(p,1) & A(p,2) & \cdots & A(p,p) \end{bmatrix}$$

Donde ***A(i,j)*** representa la distancia euclidiana entre los elementos ***M(i)*** y ***M(j).***

4. Calcular las distancias entre las muestras y los demás elementos y conformar la matriz ***B*** de la siguiente manera:

$$\begin{bmatrix} B(1,1) & B(1,2) & \cdots & B(1,r) \\ B(2,1) & B(2,2) & \cdots & B(2,r) \\ \vdots & \vdots & \ddots & \vdots \\ B(p,1) & B(p,2) & \cdots & B(p,r) \end{bmatrix}$$

Donde ***B(i, j)*** representa la distancia euclidiana entre los elementos ***M(i)*** y ***R(j)***.

[8] Aproximadamente 0.5% del total de elementos de ***C***.

5. Con las matrices A y B computar los autovectores y autovalores aproximados de la matriz laplaciana por medio del siguiente algoritmo[9].

```
d1              = sum([A;B'],1);
d2               = sum(B,1) + sum(B',1)*pinv(A)*B;
d               = [d1 d2]';
dhat            = sqrt(1./d);
A               = A.*(dhat(1:n_samp)*dhat(1:n_samp)');
B               = B.*(dhat(1:n_samp)*dhat(n_samp+(1:m))');
Asi             = sqrtm(pinv(A));
Q               = A+Asi*B*B'*Asi;
[U,L,junk]  = svd(Q);
V               = [A;B']*Asi*U*pinv(sqrt(L));       %Autovectores
for i = 1:n_samp-1
    E(:,i)    = V(:,i+1)./V(:,1);                   %Autovalores
end;
```

[9] Por comodidad en la notación el algoritmo está expresado en código de Matlab.

2.8 Histogramas[10]

2.8.1 Histogramas unidimensionales

Se utilizan para caracterizar imágenes en escala de grises (representadas por una matriz cuyos valores se encuentran en el intervalo [0, 255] o [0, 1] en el caso normalizado) o para tratar independientemente los canales RGB de una imagen con colores. Un histograma unidimensional es un vector que nos indica cuantos píxeles están dentro de determinados rangos de intensidad. Los histogramas se pueden calcular sobre toda la imagen o sobre una región de esta.

Para obtener una medida de la textura alrededor de un píxel, podemos calcular un histograma sobre una ventana alrededor de este, observe la **figura 8**:

figura 8

Para calcular un vector que caracterice el píxel central del recuadro rojo, es necesario calcular un histograma sobre dicho recuadro que permita obtener la información acerca de cuantos píxeles hay en cada posible nivel de intensidad.

[10] Los histogramas se utilizarán como vectores que caracterizan un píxel para poder clasificarlo fácilmente.

En la **figura 9** observamos una ampliación del recuadro con los valores de intensidad para cada píxel

figura 9

La **figura 10** muestra la gráfica del histograma calculado

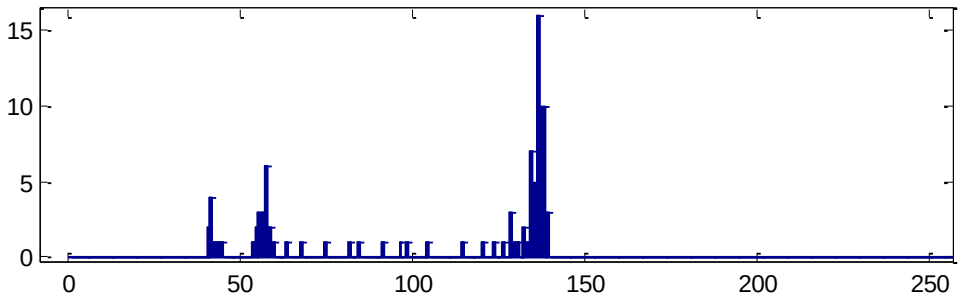

figura 10

Los elementos del vector que caracterizará el recuadro están constituidos por el número de píxeles que hay en cada posible nivel de intensidad. Para poder hacer eficientemente el mismo proceso sobre todos los píxeles de la imagen se hace necesario reducir el número de niveles de intensidad para trabajar con vectores más pequeños:

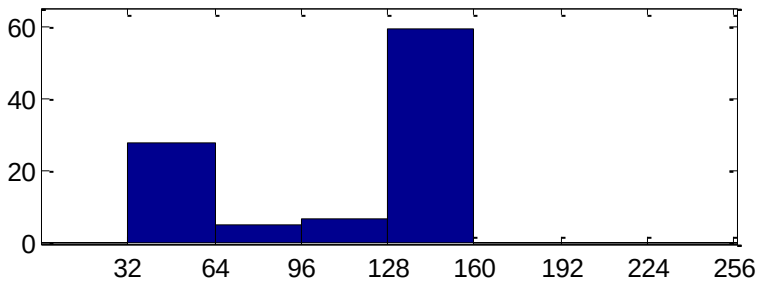

figura 11

2.8.2 Histogramas tridimensionales

Cuando se trabaja con gráficas RGB cada píxel contiene tres valores, de esta manera es posible ubicar espacialmente cada píxel en un espacio tridimensional cuyos ejes sean respectivamente los valores R, G y B. Siendo así para poder caracterizar una imagen se divide el espacio RGB en niveles que espacialmente corresponden a cubos y el histograma proporciona un vector cuyos elementos me indican cuantos píxeles hay dentro de cada cubo. En el caso unidimensional el máximo número de valores que puede tomar un píxel es de 256, en el caso tridimensional surge el problema de que el máximo número es de 16777216 posibles valores[11], pero como se vio en el caso unidimensional el número de niveles se puede reducir (**figura 11**), si reducimos el número de niveles a 8 por cada canal de color obtenemos un histograma de 512 cubos.

El histograma en tres dimensiones tiene un costo computacional mayor que el histograma unidimensional y por eso es poco recomendable.

[11] 256 valores por cada canal de color R, G y B, 256*256*256 = 16777216.

2.9 Información de contornos[12]

En muchos casos la información acerca de la intensidad o el color en una imagen no son suficientes para segmentarla en regiones coherentes. La información que proporcionan los bordes de una imagen es de gran importancia en el proceso de segmentación, el problema con los detectores de bordes tradicionales es que se basan en procesos locales, en el presente trabajo se utilizará una medida de similaridad entre píxeles a partir de *energías de orientación*[13]. Sea $\boldsymbol{F_1}$ la segunda derivada de un kernel gausiano elongado y $\boldsymbol{F_2}$ la transformada de Hilbert de $\boldsymbol{F_1}$, o sea:

$$F_1(x,y) = \frac{d^2}{dy^2}\left(\frac{1}{C}\exp\left(\frac{y^2}{\sigma^2}\right)\exp\left(\frac{x^2}{\lambda^2\sigma^2}\right)\right) \qquad \textbf{Ecuación (7)}$$

$$F_2(x,y) = Hilbert\left(F_1(x,y)\right) \qquad \textbf{Ecuación (8)}$$

Donde sigma es la escala y lamda es la elongación del filtro. C es una constante. La energía de orientación en ángulo cero está definida como:

$$OE_{0^o} = \left(I * F_1\right)^2 + \left(I * F_2\right)^2 \qquad \textbf{Ecuación (9)}$$

$\boldsymbol{OE_{0^o}}$ tiene máxima respuesta para contornos horizontales. Copias rotadas de los dos kernels tienen la capacidad de detectar bordes en diferentes orientaciones. En cada píxel se puede definir la energía de orientación y la orientación como:

---

[12] Definiciones tomadas de [16]. La información de contornos se utiliza como una alternativa para caracterizar píxeles y poder clasificarlos.

[13] ver [19] y [20]

$$OE_{con}(x,y) = \max_{\phi} OE_{\phi}(x,y)$$ ***Ecuación (9)***

$$\phi(x,y) = \arg\max_{\phi} OE_{\phi}(x,y)$$ ***Ecuación (10)***

La orientación de energía definida en las ecuaciones ***(9)*** y ***(10)*** tiene la ventaja de no ser afectada por variaciones lineales de intensidad que suelen ser causadas por sombreado suave y no por la presencia de bordes además contornos largos producirán respuestas más fuertes.

A partir de la energía de orientación se puede computar la disimilaridad entre dos píxeles basada en la existencia de bordes entre ellos, en otras palabras, dos píxeles pertenecen a regiones diferentes si existe un borde entre ellos. Siendo así la matriz de similaridad basada en contornos se calcula por medio de la siguiente fórmula:

$$W_{ij}^{IC} = \exp\left(-\frac{\max_{x \in M_{ij}} OE(x)}{\sigma_{IC}}\right)$$ ***Ecuación (10)***

Donde $\boldsymbol{M_{ij}}$ es el conjunto de máximos locales a lo largo de la línea que une 2 píxeles. Dicho de otra forma, dos píxeles tendrán una similaridad débil si existe un fuerte máximo local de energía de orientación en la línea que los une. El valor de sigma debe ser calculado experimentalmente y puede variar de acuerdo al tipo de imágenes. En este trabajo se utilizará $\boldsymbol{sigma_{contornos} = 0.15}$.

3 MARCO TEÓRICO

Las técnicas de spectral clustering fueron aplicadas inicialmente en segmentación de movimientos por Boult y Brown [1]. Estos autores proponen una restricción en el rango de la *matriz de similaridad* para estimar el número de movimientos independientes y obtener la segmentación de la imagen a partir de los principales valores singulares de dicha matriz.

Una técnica similar fue propuesta por Scott y Longuet-Higgins [5] en el contexto de segmentación de características. Estos autores asumen que el número de objetos *n* está dado y utilizan los primeros *n* autovectores de la matriz de similaridad *S* para construir una matriz de similaridad *Q* tal que $Q_{ij} = 1$ si los píxeles pertenecen al mismo objeto, de otra manera $Q_{ij} = 0$. Posteriormente la misma técnica fue aplicada por Costeira y Kanade [2] para segmentación ortográfica de movimiento[14].

Un método parecido a los anteriores pero basado en técnicas de selección de modelo fue propuesto por Kanatani [3].

Shi y Malik [23] propusieron la optimización del *corte normalizado*, una técnica que toma en cuenta tanto la medida de disimilaridad como la de similaridad entre las características asociadas a los píxeles de la imagen. Esto como una alternativa al *corte* tradicional que solo toma en cuenta la medida de la disimilaridad y tiende a proporcionar resultados erróneos. La solución óptima al corte normalizado se obtiene a partir de el segundo autovector más pequeño obtenido de una variación de la matriz laplaciana del grafo asociado a la imagen, la desventaja de este procedimiento es que solo divide la imagen en 2 regiones a la vez y esto implica que hay que aplicar el algoritmo en cada nueva subregión hasta lograr la segmentación deseada.

[14] La segmentación ortográfica de movimiento involucra imágenes obtenidas por dos o tres cámaras de video ortogonales entre sí.

Weiss [28] mostró que los algoritmos de segmentación por autovectores en [6, 21, 22, 23] tienen un alto grado de equivalencia, en algunos casos especiales también analizó las condiciones bajo las cuales ellos darían una buena segmentación. Weiss [27] posteriormente propuso un algoritmo para obtener información de la partición correcta de la imagen utilizando simultáneamente varios autovectores apoyándose en la teoría de perturbación de matrices.

Malik, Belongie y otros [9] propusieron el método de Nyström para reducir el costo computacional en el cálculo de la matriz de similaridad y sus respectivos autovectores. Este método permite -como se mencionó en el marco conceptual- calcular los autovectores de una matriz utilizando solo unas cuantas muestras aleatorias de sus elementos.

Shi, Malik y otros [15, 16] integran análisis de textura y de contornos en la extracción de características de la imagen para luego segmentarla por medio de spectral clustering. Lo anterior para poder tratar con imágenes que incluyen mosaicos de texturas.

Por último, Vidal [26] propone un método llamado segmentación polinomial y resuelve con ciertas restricciones el problema de la inicialización de algoritmos de segmentación cuando se tiene información acerca del nivel de ruido de la imagen

4 SPECTRAL CLUSTERING

La relación global entre las características de una imagen puede ser representada de manera efectiva por un grafo *completo ponderado no dirigido* cuyos vértices representan las características asociadas a cada píxel y cuyos bordes representan la medida de la similaridad entre estos. Siendo así, el hecho de lograr una partición coherente de un grafo implica también lograr una partición coherente de las regiones de la imagen asociada a dicho grafo[15]. Según la teoría espectral de grafos los autovalores (espectro) y autovectores de la matriz laplaciana asociada a un grafo proveen una base sólida para lograr una buena partición de sus vértices.

Varias técnicas basadas en el anterior principio has sido propuestas [1, 6, 13, 21, 22, 23, 27, 34], en el presente trabajo se utilizará una modificación la técnica propuesta por Weiss [27] implementando la aproximación de Nyström [9] para el cálculo de los autovectores de la matriz de similaridad. A lo largo de este capítulo se describirá el tratamiento matemático a realizarse sobre una imagen asumiendo que se conocen el número de regiones en que se segmentará. En los diferentes algoritmos propuestos se asociarán a cada píxel las siguientes características:

- Intensidad.
- Textura basada en intensidad por medio de histogramas unidimensionales.
- Color RGB.
- Textura basada en color por medio de histogramas unidimensionales.
- Textura basada en color por medio de histogramas tridimensionales.
- Contornos.
- Contornos y textura basada en histogramas unidimensionales.

[15] En general las características de un píxel que se asocian a un nodo del grafo están representadas por un vector (calculado con base en histogramas, contornos o coordenadas RGB) excepto en el caso de que la característica asociada sea la intensidad que es un escalar.

Los siguientes son los pasos generales del algoritmo de spectral clustering sin incluir el método de Nyström:

1. Conformar una matriz de características[16] en la que el elemento ***(i, j)*** corresponda al vector de características asociado al píxel ***I(i ,j)***. Transformar la matriz en el ***vector C*** encadenando sus columnas de la siguiente manera asumiendo una imagen de tamaño ***m x n*** píxeles:

$$\begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mn} \end{bmatrix} \rightarrow \begin{bmatrix} c_{11} \\ c_{21} \\ \vdots \\ c_{m1} \\ c_{12} \\ c_{22} \\ \vdots \\ c_{m2} \\ \vdots \\ \vdots \\ c_{1n} \\ c_{2n} \\ \vdots \\ c_{mn} \end{bmatrix}$$

2. Determinar el número ***k*** de regiones en que se dividirá la imagen.

3. Calcular la matriz de similaridad ***S*** a partir del ***vector C***.

[16] Cada elemento de la matriz es un vector calculado con base en el valor del píxel o su vecindad.

4. Calcular la matriz diagonal ***D*** en la cual ***D(i,i)*** es igual a la sumatoria de los elementos de la fila **i** de ***S***.

5. A partir de ***S*** y ***D*** calcular la matriz laplaciana $\boldsymbol{L = D^{-0.5} * S * D^{-0.5}}$.

6. Calcular los autovectores $\boldsymbol{x_1\ x_2 \ldots x_k}$ de ***L*** correspondientes a los ***k*** mayores autovalores y formar la matriz $\boldsymbol{X = [x_1\ x_2 \ldots x_k]}$ organizando los autovectores en columnas.

7. Obtener la matriz ***Y*** normalizando cada fila de ***X*** para que tenga magnitud unitaria por medio de la siguiente fórmula:

$$Y(i,j) = \frac{X(i,j)}{\left(\sum_j X^2(i,j)\right)^{\frac{1}{2}}}$$

8. Tratar cada fila de ***Y*** como un punto en un espacio ***k***-dimensional y agrupar dichos puntos por medio de *K-means* en ***k*** grupos. *K-means* retorna un vector ***G*** de tal forma que el valor de ***G(i)*** indica a que grupo o región corresponde el elemento ***C(i).***

9. Transformar el vector ***G(i)*** en una *matriz de etiquetas* ***E*** con las dimensiones de la imagen original ordenando los elementos por columnas:

$$G = \begin{bmatrix} G_1 \\ G_2 \\ \vdots \\ \vdots \\ G_{mxn} \end{bmatrix} \rightarrow \begin{bmatrix} G_{11} & G_{12} & \cdots & G_{1n} \\ G_{21} & G_{22} & \cdots & G_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ G_{m1} & G_{m2} & \cdots & G_{mn} \end{bmatrix} \rightarrow E = \begin{bmatrix} E_{11} & E_{12} & \cdots & E_{1n} \\ E_{21} & E_{22} & \cdots & E_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ E_{m1} & E_{m2} & \cdots & E_{mn} \end{bmatrix}$$

10. El elemento ***E(i,j)*** indica a que región o grupo pertenece el píxel ***I(i, j).***

Para una imagen de ***N*** píxeles, la matriz ***S*** contiene N^2 elementos, esto representa un alto costo computacional que a su vez implica un alto consumo de tiempo de ejecución, para evitar esto se implementará el método de Nyström que reduce notablemente el tiempo requerido para los pasos 3, 4, 5 y 6.

El siguiente es el algoritmo de spectral clustering modificado para incluir el método de Nyström:

1. Conformar una matriz de características en la que el elemento ***(i,j)*** corresponda al vector de características asociado al píxel ***I(i,j)***. Transformar la matriz en el ***vector C*** encadenando sus columnas de la siguiente manera asumiendo una imagen de tamaño ***m x n*** píxeles:

$$\begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mn} \end{bmatrix} \rightarrow \begin{bmatrix} c_{11} \\ c_{21} \\ \vdots \\ c_{m1} \\ c_{12} \\ c_{22} \\ \vdots \\ c_{m2} \\ \vdots \\ \vdots \\ c_{1n} \\ c_{2n} \\ \vdots \\ c_{mn} \end{bmatrix}$$

2. Determinar el número ***k*** de regiones en que se dividirá la imagen.

3. Calcular las matrices de distancias ***A*** y ***B*** a partir del vector ***C.***

4. Aplicar el método de Nyström para obtener los autovectores y autovalores de la matriz laplaciana.

5. Tomar los autovectores $x_1\ x_2 \ldots x_k$ correspondientes a los ***k*** mayores autovalores y formar la matriz $X = [x_1\ x_2 \ldots x_k]$ organizando los autovectores en columnas.

6. Obtener la matriz ***Y*** renormalizando cada fila de ***X*** para que tenga magnitud unitaria por medio de la siguiente fórmula:

$$Y(i, j) = \frac{X(i, j)}{\left(\sum_j X^2(i, j) \right)^{\frac{1}{2}}}$$

7. Tratar cada fila de ***Y*** como un punto en un espacio ***k***-dimensional y agrupar dichos puntos por medio de ***K-means*** en ***k*** grupos. K-means retorna un vector ***G*** de tal forma que el valor de ***G(i)*** indica a que grupo o región corresponde el elemento ***C(i).***

8. Transformar el vector ***G(i)*** en una *matriz de etiquetas* ***E*** con las dimensiones de la imagen original ordenando los elementos por columnas:

$$G = \begin{bmatrix} G_1 \\ G_2 \\ \vdots \\ \vdots \\ G_{m\,x\,n} \end{bmatrix} \rightarrow \begin{bmatrix} G_{11} & G_{12} & \cdots & G_{1n} \\ G_{21} & G_{22} & \cdots & G_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ G_{m1} & G_{m2} & \cdots & G_{mn} \end{bmatrix} \rightarrow E = \begin{bmatrix} E_{11} & E_{12} & \cdots & E_{1n} \\ E_{21} & E_{22} & \cdots & E_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ E_{m1} & E_{m2} & \cdots & E_{mn} \end{bmatrix}$$

9. El elemento ***E(i,j)*** indica a que región o grupo pertenece el elemento ***I(i,j).***

4.1 Spectral clustering aplicado a imágenes en escala de grises

4.1.1 Agrupamiento por valores de intensidad[17]

Para desarrollar esta implementación se utilizarán las siguientes imágenes:

figura 12

A la izquierda está la imagen original, a la derecha está la imagen que será procesada por el algoritmo. Las imágenes tienen un tamaño de 130 x 180 píxeles para un total de ***N = 23400*** píxeles. La característica que se asociará a cada píxel será su valor de intensidad de esta manera la matriz de características es la misma matriz que representa a la imagen de la derecha. La imagen será dividida en ***k = 2*** regiones.

Una vez obtenidos los ***2*** mayores autovectores x_1 y x_2 por medio del método de Nyström creamos la matriz ***X = [x1 x2]*** y asumiendo que cada fila es un punto en un espacio 2-dimensional agrupamos todos los puntos por medio de K-means en 2 regiones.

De esta manera podemos saber a que región pertenece cada píxel de la imagen original. Visualizando la matriz de etiquetas podemos ver claramente las regiones resultantes. En la **figura 13** observamos las dos regiones y la matriz de etiquetas.

[17] La explicación general del algoritmo en código de Matlab se encuentra en el último capítulo.



figura 13
a. Región 1. **b.** Región 2 **c.** Matriz de etiquetas

Ficha técnica

Formato	Escala de grises
Tamaño	130 x 180
Tiempo de ejecución	Calculando matrices A y B 0.301000 segundos Aproximación de Nyström 1.332000 segundos Ejecutando k-means 0.120000 segundos Graficando 0.400000 segundos Total: 2.684000 segundos.
Valor de sigma	2.5
Número de muestras para la aproximación de Nyström.	30 de un total de 23400

Sin la ayuda del método de Nyström el tiempo total de ejecución sobrepasaría los 4 minutos, el valor de sigma determina que tan rápido decaen las afinidades entre los píxeles y a lo largo de este trabajo se utilizará casi siempre el mismo valor excepto cuando sea necesario incrementarlo para poder discriminar más fácilmente determinadas texturas. A continuación apreciaremos los resultados de procesar la misma imagen con sigma = 1, 1.5, 2, 2.5, 4.

En la figura anterior se aprecia que valores de sigma menores de 2.5 no logran una buena discriminación, lograr un ajuste automático del valor de sigma requiere ejecutar varias veces el algoritmo y realizar un análisis de qué tan compactas resultan las regiones, experimentalmente 2.5 ha proporcionado buenos resultados en la mayoría de los casos. Ahora se realizará la segmentación de una imagen en un número diferente de regiones.

figura 15
Figura original y en escala de grises. Observando la imagen original se espera una segmentación de cuatro regiones, el fondo, el cabello, la piel y la ropa. Sin embargo se aprecia que la pérdida del color hace que el número de regiones disminuya ya que la piel y la ropa tienen niveles de intensidad similares.

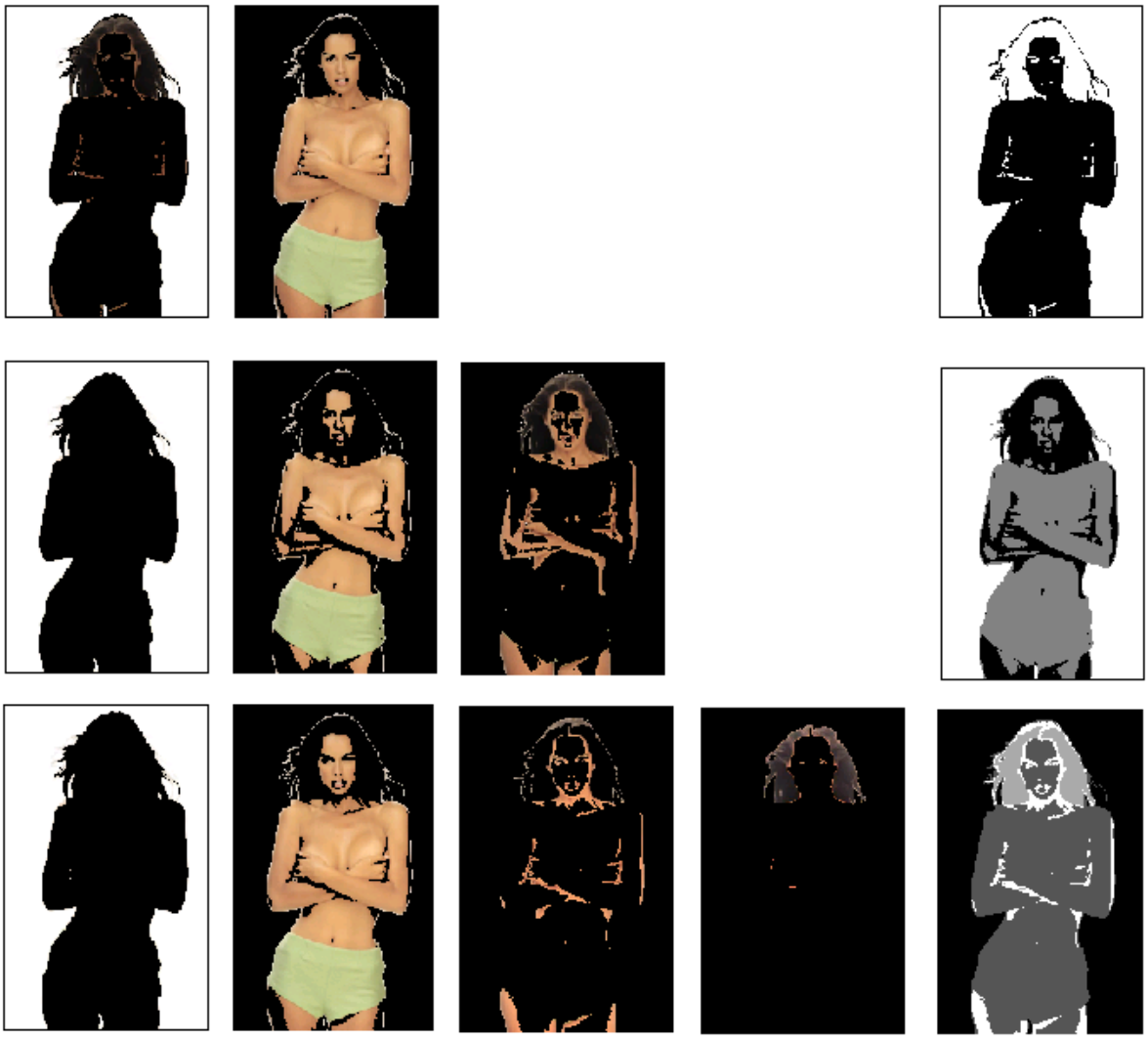

figura 16
Segmentación en 2, 3 y 4 regiones, se aprecia que la información de intensidad no es suficiente para discriminar satisfactoriamente 4 grupos, sin embargo las regiones obtenidas son relativamente compactas.

El anterior algoritmo resulta efectivo solamente cuando no existen texturas complicadas y se diferencian claramente objetos con intensidades diferentes, esto lo hace útil en pocas aplicaciones prácticas sin embargo puede ser una alternativa para determinados casos. En las siguientes figuras se observa un ejemplo con una imagen para la cual el algoritmo es efectivo y una para la que falla siempre.

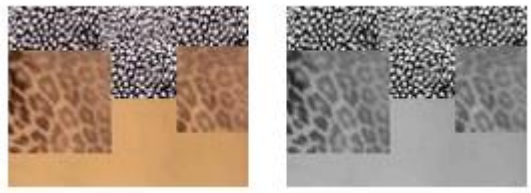

figura 17
Imagen original y en escala de grises.

figura 18
Segmentación en 3 regiones y matriz de etiquetas de la figura 17, se observa que el algoritmo falla cuando trata de procesar texturas, en la matriz de etiquetas deberían aparecer cuatro regiones compactas. La región sin textura es la única compacta.

figura 19
Imagen original y en escala de grises.

figura 20
Segmentación en 3 regiones y matriz de etiquetas de la figura 19. Se observa que el algoritmo es relativamente eficiente en ausencia de texturas y proporciona regiones compactas.

4.1.2 Agrupamiento por textura en escala de grises[18]

Para desarrollar este caso se utilizarán inicialmente las siguientes imágenes:

figura 21
Imagen original y en escala de grises.

Se escogió un valor de k = 4 regiones (el fondo, el cabello, el rostro y la ropa).

Para conformar el vector de características se asocia a cada píxel un histograma calculado sobre una ventana que lo contiene. Las dimensiones de la ventana pueden variar pero hay que tener en cuenta que una ventana muy pequeña o muy grande no caracterizará muy bien al píxel. Un valor de 7 x 7 o de 9 x 9 píxeles entrega buenos resultados[19]. El valor del lado de la ventana siempre es impar para poder ubicar el píxel de interés en el centro de esta. El histograma utilizado informa sobre la cantidad de píxeles que están dentro de los siguientes rangos de intensidad:

[0 16 32 48 64 80 96 112 128 144 160 176 192 208 224 240 255]

Proporcionando un vector de 16 elementos que nos informa acerca de la textura alrededor del píxel.

---

[18] La explicación detallada del algoritmo en código de Matlab se encuentra en el último capítulo.
[19] Encontrar un valor adecuado de forma automática para el tamaño de la ventana es aún motivo de investigación en la comunidad científica y está fuera del alcance de este trabajo, los valores aquí propuestos son los que en la práctica mejores resultados han dado.

Una vez conformado el vector de características hay que proceder con el cálculo de las matrices A y B.

Ya que las distancias se calculan entre vectores y no entre escalares como en el caso de segmentación por intensidad el costo computacional es mayor. A partir del agrupamiento sobre los autovectores hallados por medio del método de Nyström se obtiene la siguiente segmentación:

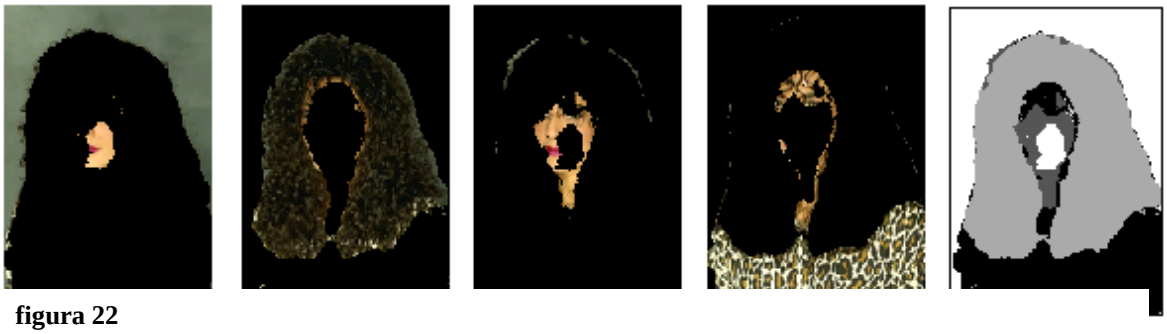

figura 22
Segmentación en 4 regiones y matriz de etiquetas de la figura 21. El algoritmo tuvo problemas con el rostro pero en términos generales fue eficiente agrupando regiones con textura de manera compacta. El algoritmo basado en intensidad habría fallado con toda seguridad.

Ficha técnica segmentación de **la figura 21**

Formato	Escala de grises
Tamaño	179 x 120
Tiempo de ejecución	Calculando matriz de histogramas 1.241000 Calculando matrices A y B 0.43000 segundos Aproximación de Nyström 1.062000 segundos Ejecutando k-means 0.18000 segundos Graficando 0.571000 segundos Total: 3.645000 segundos.
Valor de sigma	2.5
Número de muestras	30 de un total de 21480

A continuación se visualizará el desempeño del algoritmo sobre diferentes imágenes.

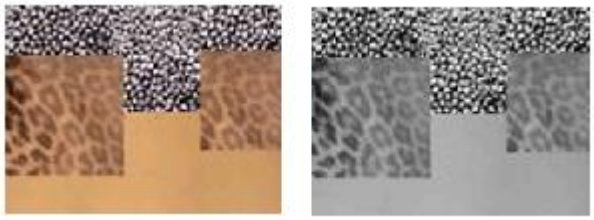

figura 23
Imagen original y en escala de grises.

figura 24
Segmentación en 3 regiones y matriz de etiquetas de las imágenes de la figura 23. El algoritmo muestra ser eficiente en regiones de textura constante proporcionando regiones compactas.

figura 25
Imagen original y en escala de grises. Se observa que el número de regiones bien diferenciadas cambia un poco con la pérdida del color.

figura 26
Segmentación en 4 regiones y matriz de etiquetas de las imágenes de la figura 25. El algoritmo no se muestra eficiente cuando las texturas aumentan su grado de similaridad al convertir la imagen a escala de grises.

figura 27
Imagen original y en escala de grises. Existe una clara pérdida de contraste entre las texturas de las diferentes regiones.

figura 28
Segmentación en 3 regiones y matriz de etiquetas de las imágenes de la figura 27. A pesar del bajo contraste de las texturas el algoritmo es relativamente eficiente en discriminar el cielo, el agua y las rocas en regiones poco compactas.

figura 29
Imagen original y en escala de grises, la pérdida del color causa una gran pérdida de contraste entre las texturas originales.

figura 30
Segmentación en 2 regiones y matriz de etiquetas de las imágenes de la figura 29. El algoritmo falla en la discriminación de texturas.

4.1.3 Agrupamiento por información de contornos

En este caso la matriz de similaridad se forma como se indico en la ecuación ***(11)*** a partir de la energía de orientación. Las características asociadas a los píxeles determinan si estos están separados entres si por contornos. Este enfoque por si solo no siempre es muy efectivo ya que suele suceder que píxeles que pertenecen a una misma región están separados por fuertes contornos.

figura 31
Imagen original, en escala de grises y gráfica de la energía de orientación calculada.

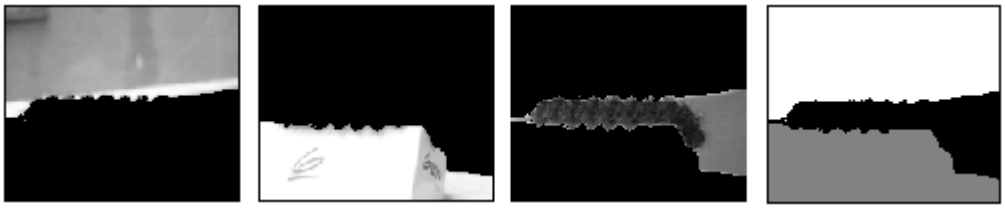

Figura 32
Segmentación en 3 regiones y matriz de etiquetas de las imágenes de la figura 31.

Ficha técnica segmentación de la figura 31

Formato	Escala de grises
Tamaño	107 x 130
Tiempo de ejecución	Calculando matriz de similaridad 4.486000 Calculando laplaciano 0.211000 segundos Calculando autovectores 4.663000 segundos Ejecutando k-means y graficando 0.6410 Total: 9.994 segundos.
Valor de sigma	2.5

figura 33
Imagen original, en escala de grises y gráfica de los máximos de la energía de orientación.

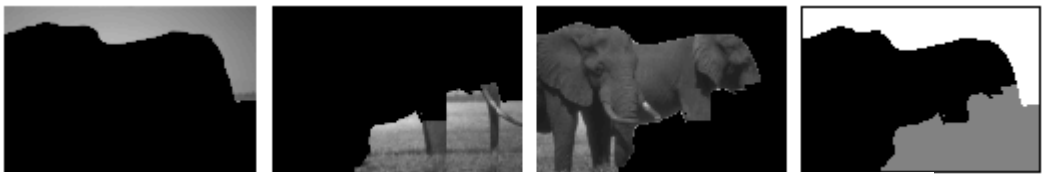

figura 34
Segmentación en 3 regiones y matriz de etiquetas de las imágenes de la figura 34

Figura 35
Imagen original, en escala de grises y gráfica de los máximos de la energía de orientación.

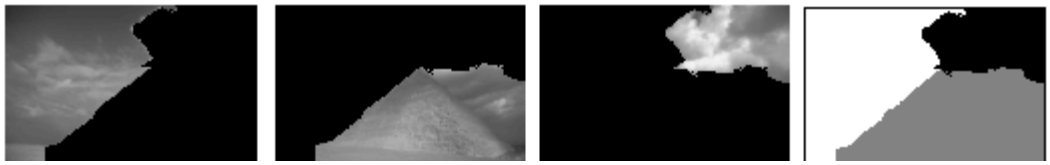

Figura 36
Segmentación en 3 regiones y matriz de etiquetas de las imágenes de la figura 35.

Los contornos proporcionan información valiosa para la segmentación pero es claro que se necesita combinarlos con otras estrategias para obtener mejores resultados, aunque esto cause un incremento en el tiempo de procesamiento.

4.2 Spectral clustering aplicado a imágenes RGB

4.2.1 Agrupamiento por valores RGB

La característica que se asocia a cada píxel es su coordenada en el espacio RGB.

figura 37
Imagen original.

figura 38
Segmentación en 2 regiones y matriz de etiquetas de la imagen de la **figura 37**. El algoritmo segmenta fácilmente regiones de color diferente.

Ficha técnica segmentación de la **figura 21**

Formato	RGB
Tamaño	120 x 180
Tiempo de ejecución	Calculando matrices A y B 0.090000 segundos Aproximación de Nyström 0.971000 segundos Ejecutando k-means 0.221000 segundos Graficando 0.37000 segundos Total: 2.433000 segundos.
Valor de sigma	2.5
Número de muestras	30 de un total de 21600

Los algoritmos de agrupamiento basados en intensidad y en color RGB son similares en cuanto a que, como veremos a continuación en la **figura 33**, no son buenos procesando texturas. Por otro lado el algoritmo RGB supera al de intensidad al discriminar regiones de colores diferentes con el mismo nivel de intensidad.

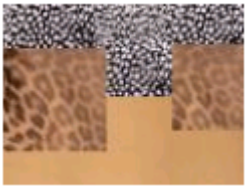

figura 39
Imagen original.

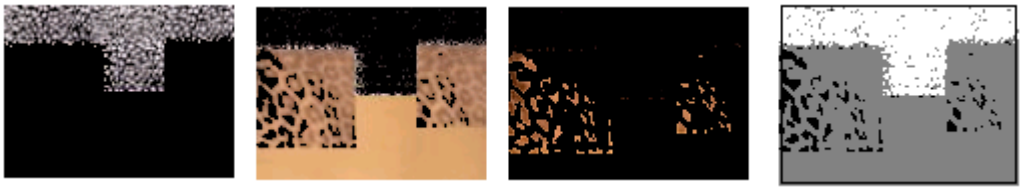

figura 40
Segmentación en 3 regiones y matriz de etiquetas de la imagen de la figura 39. El algoritmo es incapaz de segmentar regiones ricas en textura.

figura 41
Imagen original.

figura 42
Segmentación en 3 regiones y matriz de etiquetas de la imagen de la figura 41. El algoritmo es eficiente diferenciando regiones de intensidad similar y diferente color.

4.2.2 Agrupamiento por histogramas RGB en tres dimensiones

Las características que se asocian a los píxeles en este caso son histogramas en tres dimensiones, el espacio RGB se divide en 512 cubos, o sea 8 niveles por cada canal de color, se calcula un histograma para cada píxel sobre una ventana de 7 x 7 píxeles. Este algoritmo es más lento a pesar de la aproximación de Nyström ya que trabaja con vectores de características de 512 elementos.

figura 43
Imagen original.

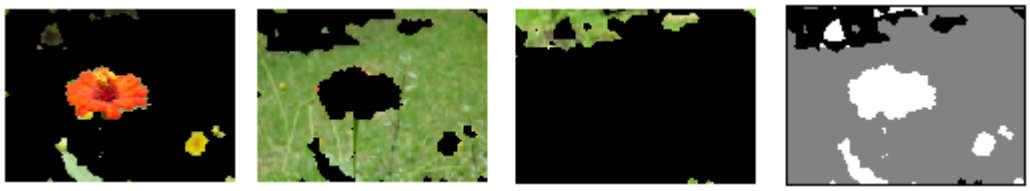

figura 44
Segmentación en 3 regiones y matriz de etiquetas de la imagen de la figura 43. El algoritmo es eficiente diferenciando regiones de textura y color diferentes.

Ficha técnica segmentación de la **figura 36**

Formato	RGB
Tamaño	90 x 120
Tiempo de ejecución	Calculando matrices A y B 4.94700 segundos Aproximación de Nyström 1.032000 segundos Ejecutando k-means 0.28000 segundos Graficando 2.193000 segundos Total: 17.395000 segundos.
Valor de sigma	2.5
Número de muestras	40 de un total de 10800

figura 45
Imagen original.

figura 46
Segmentación en 3 regiones y matriz de etiquetas de la imagen de la figura 45.

figura 47
Imagen original.

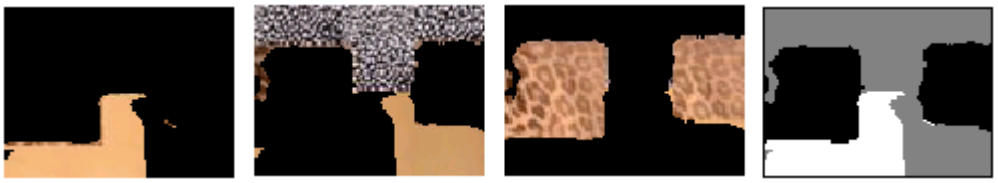

figura 48
Segmentación en 3 regiones y matriz de etiquetas de la imagen de la figura 47.

En las figuras anteriores se observa que el algoritmo falla en la discriminación de regiones coherentes, esto se debe a que el espacio RGB está dividido en 512 cubos y esto no siempre es suficiente, lo ideal sería dividir cada canal de color en 16 o sea dividir el espacio RGB en 4096 cubos pero esto no es viable en términos computacionales.

4.2.3 Agrupamiento por histogramas RGB unidimensionales

Una imagen RGB está compuesta de tres matrices, una para cada canal de color, para este algoritmo se calcularán histogramas unidimensionales alrededor de cada píxel de forma independiente en cada canal de color, el vector característico de cada píxel estará conformado por la concatenación de estos tres histogramas cada uno de los cuales clasifica los píxeles en 16 niveles de intensidad, de esta forma el vector de características posee 48 elementos. Esto permite extraer características de textura y color sin necesidad de computar histogramas tridimensionales. La ventana alrededor de cada píxel es de 7x7.

figura 49
Imagen original.

figura 50
Segmentación en 3 regiones y matriz de etiquetas de la imagen de la figura 49. Discriminando texturas en color este algoritmo es más eficiente que el que utiliza histogramas tridimensionales.

Formato	RGB
Tamaño	86 x 130
Tiempo de ejecución	Calculando matrices A y B 0.771000 segundos Aproximación de Nyström 3.495000 segundos Ejecutando k-means 0.48000 segundos Graficando 0.44000 segundos Total: 7.691000 segundos.
Valor de sigma	2.5
Número de muestras	40 de un total de 11180

figura 51
Imagen original.

figura 52
Segmentación en 4 regiones y matriz de etiquetas de la imagen de la figura 51. El algoritmo no es muy eficiente con los contornos y discrimina mejor algunas texturas que otras.

figura 53
Imagen original.

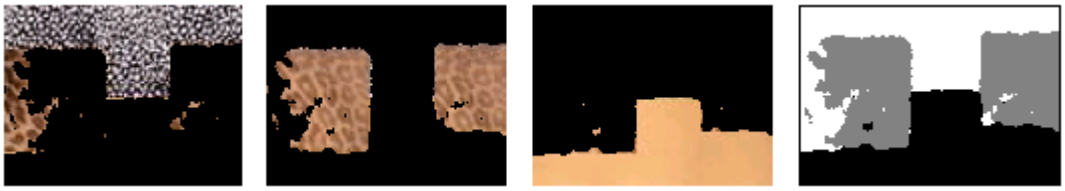

figura 54
Segmentación en 3 regiones y matriz de etiquetas de la imagen de la figura 53. El algoritmo no es muy eficiente con los contornos pero discrimina bien las diferentes texturas.

En las figuras anteriores se aprecia que en términos generales el algoritmo funciona muy bien, sin embargo no es eficiente en los bordes.

4.3 Agrupamiento por contornos e histogramas unidimensionales en color

En este caso combinaremos la información de los autovectores obtenidos en el algoritmo basado en contornos con los autovectores obtenidos en el algoritmo basado en histogramas de color unidimensionales. Dicho de otra forma, siendo $\boldsymbol{X_{ic}}$ la matriz cuyas columnas contienen los autovectores normalizados basados en contornos y $\boldsymbol{X_{hc}}$ la matriz cuyas columnas contienen los autovectores normalizados basados en histogramas unidimensionales en color, entonces aplicamos k-means sobre la matriz:

$$\boldsymbol{X_{ic\_hc} = [Xic, H_{hc}]}$$

De esta manera la agrupación toma en cuenta simultáneamente los dos tipos de información, esto es muy provechoso ya que el algoritmo basado solamente en textura de color no detecta regiones de color similar separadas por uno o varios contornos.

En general se puede combinar información de autovectores calculados con base en cualquier característica, las posibles combinaciones son demasiadas para ser tratadas en este documento.

Combinar información de dos características diferentes tiene un par de inconvenientes, en primer lugar el incremento en el tiempo de procesamiento y en segundo lugar posibles conflictos ya que es posible que exista alguna región que posea contornos pero no deba ser dividida. En general la combinación de características que se extraen de la imagen tiene más ventajas que desventajas cuando se trata con imágenes ricas en textura y color.

A continuación se presentan algunos ejemplos sobre imágenes nuevas y sobre imágenes ya procesadas con otros algoritmos.


figura 55
Imagen original.



figura 56
Segmentación en 3 regiones y matriz de etiquetas de la imagen de la figura 55. El algoritmo muy eficiente con los contornos y discrimina bien las diferentes texturas.


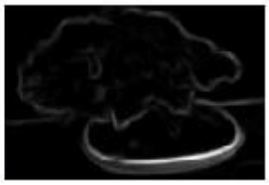

figura 57
Imagen original.



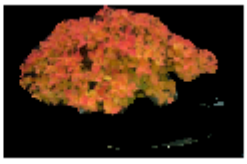


figura 58
Segmentación en 3 regiones y matriz de etiquetas de la imagen de la figura 57. El algoritmo muy eficiente con los contornos y discrimina bien las diferentes texturas.


figura 59
Imagen original.




figura 60
Segmentación en 3 regiones y matriz de etiquetas de la imagen de la figura 59. El algoritmo muy eficiente con los contornos y discrimina bien las diferentes texturas y colores.

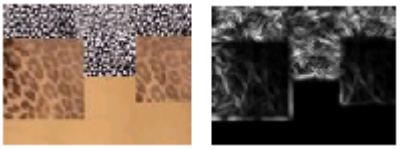

figura 61
Imagen original.

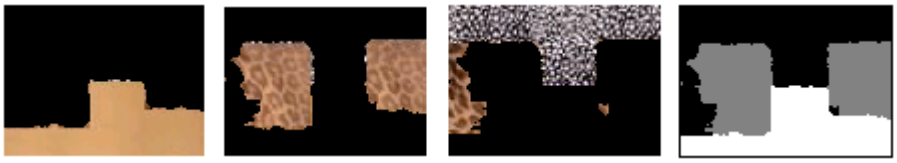

figura 62
Segmentación en 3 regiones y matriz de etiquetas de la imagen de la figura 61. El algoritmo muy eficiente con los contornos y discrimina bien las diferentes texturas y colores en imágenes sintéticas.

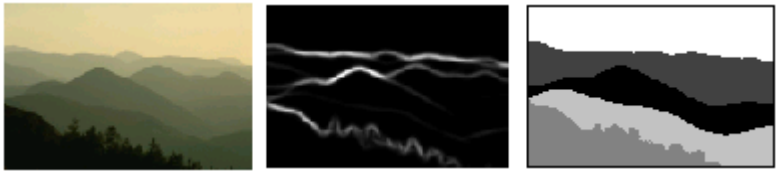

figura 63
Imagen original.

figura 64
Segmentación en 3 regiones y matriz de etiquetas de la imagen de la figura 63. El algoritmo muy eficiente con los contornos y discrimina texturas y colores similares.

Formato	RGB
Tamaño	100 x 150
Tiempo de ejecución	Calculando autovectores basados en contornos 17.5250 segundos Calculando matriz de histogramas 1.2420 segundos Calculando matrices A y B 0.5000 segundos Aproximación de Nyström 0.832 segundos Ejecutando k-means 0.25000 segundos Graficando 2.374 segundos Total: 22.863 segundos.
Valor de sigma	2.5
Número de muestras	40 de un total de 15000

5 SEGMENTACIÓN POLINOMIAL[20]

La segmentación polinomial es un método que permite calcular el número ***n*** de grupos en que se debe segmentar una imagen de acuerdo a sus niveles de intensidad[21], hasta el momento para utilizar los algoritmos de spectral clustering el número ***n*** de grupos se ha escogido manualmente y básicamente lo que se ha hecho es lograr que a cada píxel le corresponda un valor de los ***n*** posibles valores que tiene la matriz de etiquetas.

La segmentación polinomial permite inicialmente encontrar los ***n*** valores que tomará la matriz de etiquetas y proporciona una fórmula para encontrar que valor le corresponde a cada píxel. Para describir el proceso primero se debe transformar la matriz de intensidades ***T*** de ***m x p = N*** píxeles en un vector columna ***X***:

$$\begin{bmatrix} T_{11} & T_{12} & \cdots & T_{1p} \\ T_{21} & T_{22} & \cdots & T_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ T_{m1} & T_{m2} & \cdots & T_{mp} \end{bmatrix} \rightarrow \begin{bmatrix} T_{11} \\ T_{21} \\ \vdots \\ T_{m1} \\ T_{12} \\ T_{22} \\ \vdots \\ T_{m2} \\ \vdots \\ \vdots \\ T_{1p} \\ T_{2p} \\ \vdots \\ T_{mp} \end{bmatrix} \rightarrow \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ x_N \end{bmatrix}$$

[20] Algoritmo propuesto en [26] y utilizado en el presente trabajo par inicializar algoritmos basados en spectral clustering.

[21] También se pueden utilizar otros criterios, por ejemplo textura, en el presente trabajo solo se implementará la segmentación polinomial basada en intensidad.

El píxel ***X(j)*** debe corresponder alguno de los posibles valores ***[I_1, I_2, . . I_n]*** de la matriz de etiquetas, dicho de otro modo:

$$\left(X(j)=I_1\right)\vee\left(X(j)=I_2\right)\vee\ldots\vee\left(X(j)=I_n\right)$$

lo que puede ser escrito como el siguiente polinomio de grado n

$$Pn(x)=\prod_{i=1}^{n}\left(X(j)-I_i\right)=\sum_{k=0}^{n}c_k x^k=0 \qquad \textbf{\textit{Ecuación (11)}}$$

Ya que la ecuación anterior es válida para todos los elementos de X, se tiene:

$$L_n c=\begin{bmatrix}1 & x_1 & x_1{}^2 & \cdots & x_1{}^n\\ 1 & x_2 & x_2{}^2 & \cdots & x_2{}^n\\ \vdots & \vdots & \vdots & \cdots & \vdots\\ 1 & x_N & x_N 2 & \cdots & x_N{}^n\end{bmatrix}\begin{bmatrix}c_0\\ \vdots\\ c_{n-1}\\ 1\end{bmatrix}=0 \qquad \textbf{\textit{Ecuación (12)}}$$

Para que el anterior sistema de ecuaciones lineales posea una solución única para el vector de coeficientes c, se necesita que ***rango(Ln) = n*** esta restricción en $\boldsymbol{L_n}$ provee un criterio para determinar el número de grupos ***n*** a partir del vector ***X*** de la siguiente manera, siendo $Li\in\Re^{Nx(i+1)}$ la matriz formada por las primeras ***i + 1*** columnas de $\boldsymbol{L_n}$, si ***N >= n***, entonces:

$$rango(Li)\begin{cases}>i, & si \quad i<n,\\ =i, & si \quad i=n,\\ <i, & si \quad i>n.\end{cases}$$

O sea que el número de grupos está dado por:

$$n = \min\{i : rango(Li) = i\} \qquad \textbf{Ecuación (13)}$$

En ausencia de ruido, es posible estimar el número de grupos de forma incremental por medio del anterior criterio. Si existe ruido esto no siempre es posible ya que las columnas de $\boldsymbol{L_i}$ podrían ser linealmente independientes para todo $\boldsymbol{i}$. En ese caso el número de grupos se determina así:

$$n = \min\left\{ i : \frac{\sigma_{i+1}}{\sigma_i} < \varepsilon \right\} \qquad \textbf{Ecuación (14)}$$

donde σ_i es el iésimo valor singular de $\boldsymbol{Li}$ y ε es un umbral preestablecido que depende del nivel de ruido[22].

Con el número de grupos ya es posible inicializar un algoritmo basado en spectral clustering, por otra parte se puede encontrar la solución al sistema de ecuaciones ***(12)*** encontrando los valores de ***c***. Con los valores de ***c*** es posible calcular los valores de las raíces $\boldsymbol{[I_1, I_2, \ldots I_n]}$ del polinomio de la ecuación ***(11)*** o sea los posibles valores para la matriz de etiquetas. Finalmente para determinar que valor $\boldsymbol{I_i}$ corresponde al píxel ***X(j)*** se utiliza la fórmula:

$$i = \min_{l=1,\ldots,n} \left(x_j - I_l \right)^2$$

[22] Las imágenes que se utilizan en este trabajo contienen un bajo nivel de ruido y no ha sido necesario utilizar este criterio.

figura 65
Imagen original y matriz de etiquetas proporcionada por el algoritmo de segmentación polinomial.

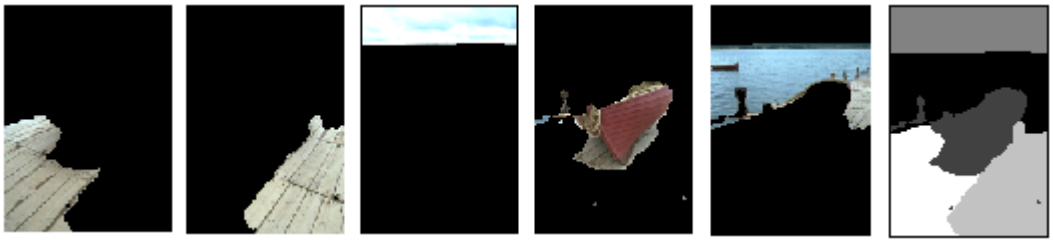

figura 66
Segmentación en 5 regiones y matriz de etiquetas de la imagen de la figura 65 por medio de spectral clustering basado en contornos e histogramas en color inicializado con segmentación polinomial, el algoritmo funciona bien pero tiende a sobreestimar el número de grupos

figura 67
Imagen original y matriz de etiquetas proporcionada por el algoritmo de segmentación polinomial.

figura 68
Segmentación en 5 regiones y matriz de etiquetas de la imagen de la figura 67 por medio de spectral clustering basado en contornos e histogramas en color inicializado con segmentación polinomial, el algoritmo discrimina bien pro sobreestima el número de regiones.

6 CONCLUSIONES

Los algoritmos de segmentación basados en spectral clustering demuestran ser de gran utilidad al permitir segmentar imágenes con base en la extracción de diferentes características. Aunque las diferentes implementaciones realizadas en el presente trabajo fueron realizadas por medio de matlab, el código podría implementarse fácilmente con lenguajes tales como C, C++ o JAVA obteniendo un mejor rendimiento y un costo menor de implementación en cuanto al costo de las licencias comerciales.

Hay que mencionar que es necesario profundizar acerca de la escogencia de algunos parámetros tales como el número de píxeles que deben contener las ventanas sobre las cuales se calculan los histogramas, el valor de sigma y el número de elementos del vector de características, esto conllevaría a una mejor relación entre la calidad de la segmentación y el tiempo de ejecución.

La mayoría de las opciones de extracción de características se muestran útiles de acuerdo a las necesidades de segmentación y a la complejidad de las imágenes, los algoritmos más rápidos son los basados en intensidad y en color RGB, si bien es cierto que no son eficientes procesando texturas resultan de gran utilidad en ambientes controlados cuando las imágenes poseen regiones con niveles de intensidad y color constantes, esto es de gran aplicabilidad en control de calidad y seguimiento de trayectorias en visión artificial.

Los algoritmos más complejos son más lentos y tienen parámetros que se pueden optimizar, de todas formas pueden ser de gran utilidad en tareas avanzadas de visión robótica y organización perceptiva.

La cantidad de información disponible acerca de extracción de características en imágenes es bastante grande, las segmentaciones basadas en intensidad, color RGB, histograma en escala de grises, histogramas tridimensionales y contornos han sido utilizados por otros

autores con resultados similares, por otro lado, la extracción de histogramas unidimensionales y su combinación con la información de contornos por medio de la concatenación de autovectores propuesta en el presente trabajo no había sido implementada antes y demuestra ser robusta en la discriminación de texturas y colores.

La técnica de segmentación polinomial a pesar de sobreestimar el número de grupos demuestra se una herramienta de vital importancia en el proceso de segmentación automática ya que permite obtener una aproximación de número inicial de grupos en que se puede dividir una imagen, con base en esto se puede calcular un número más exacto por medio de un post-procesamiento basado en la comparación de curvas de convergencia para diferentes números de regiones de algoritmos de agrupación como k-means.

7 CÓDIGO FUENTE

Nys_AB_euc.m

```
% function [ind_all, A, B] = nys_AB_euc(data,sigma,nsamp)
% Argumentos de entrada
%       data:       Matriz de la cual se van a calcular los autovecores
%                   cada fila corresponde a una carácterística.
%       sigma:      Valor que indica que tan rápido decae la afinidad
%                   entre las características dependiendo de la distancia
%                   entre ellas.
%       nsamp:      Número de muestras para realizar la aproximación de
%                   Nystrom
%
% Argumentos de Salida
%       ind_all:    Vector con los índices de las filas de la matriz data
%                   en orden aleatorio.
%       A:          Matriz que contiene las distancias por pares
%                   entre las muestras seleccionadas.
%       B:          Matriz que contiene las distancias entre las muestras
%                   seleccionadas y el resto de los puntos en la matriz
%                   data.
% pairwiseSqrDist es un mex file que calcula la matriz de distancias
% cuadaradas
% John Peña.  Diciembre de 2004.



function [ind_all, A, B] = nys_AB_euc(data,sigma,nsamp)


[fil,col,pro]   = size(data);
ind_all         = randperm(fil);
ind_samp        = ind_all(1:nsamp);
ind_rest        = ind_all(nsamp+1:end);


dist_sam        = pairwiseSqrDist(data(ind_samp,:)',data(ind_samp,:)');
dist_sam_rest   = pairwiseSqrDist(data(ind_samp,:)',data(ind_rest,:)');

A               = exp(-(dist_sam)/(2*sigma^2));
B               = exp(-(dist_sam_rest)/(2*sigma^2));
```

Nys_meth.m

```
% Calcula los autovectores de la matriz laplaciana por medio de las
% matrices A y B


function [E,val]=nys_meth(A,B);

n_samp      = size(A,1);    %número de muestras
m           = size(B,2);    %número de datos - número de muestras
nu_pix      = n_samp + m;   %numero de datos (pixels)



d1          = sum([A;B'],1);
d2          = sum(B,1) + sum(B',1)*pinv(A)*B;

disp('Normalizando...');
d           = [d1 d2]';
dhat        = sqrt(1./d);
A           = A.*(dhat(1:n_samp)*dhat(1:n_samp)');
B           = B.*(dhat(1:n_samp)*dhat(n_samp+(1:m))');
disp('Calculando autovectores... ');

Asi         = sqrtm(pinv(A));
Q           = A+Asi*B*B'*Asi;
%[U,S,junk]  = svd(A);
%Asi         = U*pinv(sqrt(S))*U';
%Q           = A+Asi*B*B'*Asi;
[U,L,junk]  = svd(Q);
V           = [A;B']*Asi*U*pinv(sqrt(L));

for i = 1:n_samp-1
  E(:,i)    = V(:,i+1)./V(:,1);
end;

val=1-diag(L);
val(1)=[];
```

sp_open.m

```
% lectura de un archivo de imagen, cambio del tamaño y transformación
% opcional en una imagen de escala de grises de 8 bits.
%
% argumetnos de entrada:
% img       : cadena que contiene el nobre del archivo
% x_new     : nuevo valor de la longitud en x de la imagen
%             el valor de la longitud en y se calcula
%             automáticamaente para conservar el aspect ratio.
% method    : método utilizado para cambiar el tamaño de la
%             imagen (puede ser 'nearest', 'bilinear' o 'bicubic').
% argumentos de salida:
% im_orig   : contiene la matriz de  datos de la imagen original
%             en caso de que la imagen original esté en formato de
%             imagen indexada esta es transformada a rgb.
%
% im_resized : estructura que contiene los datos de la imagen en grises


function [im_resized, im_orig]=sp_open(img, x_new, method, color)

    im_info     = imfinfo(img);
    if(strcmp(im_info.ColorType,'truecolor'))
        im_orig         = imread(img);
        im_new          = im_orig;
        if(~color)
            im_new      = rgb2gray(im_orig);
        end
        im_resized      = (sp_resize(im_new, x_new, method));
        im_orig         = sp_resize(im_orig, x_new, method);
    end
    if(strcmp(im_info.ColorType,'indexed'))
        [im, map]       = imread(img);
        im_orig         = ind2rgb(im,map);
        im_new          = im_orig;
        if(~color)
            im_new      = ind2gray(im, map);
        end
        im_resized      = (sp_resize(im_new, x_new, method));
        im_orig         = sp_resize(im_orig, x_new, method);
    end
        if(strcmp(im_info.ColorType,'grayscale'))
        im_orig         = imread(img);
        im_resized      = (sp_resize(im_orig, x_new, method));
        im_orig         = sp_resize(im_orig, x_new, method);
    end
```

sp_resize.m

```
% función sp_resize
% cambia el tamaño de una imagen conservando el aspect ratio.
%
% argumentos de entrada:
% im_orig       : imagen original
% x_new         : nuevo valor de la longitud en x de la %
        imagen el valor de la longitud en y se
%                    calcula automáticamaente para conservar el %
        aspect ratio.
% method        : método utilizado(puede ser 'nearest',
%                   'bilinear' o 'bicubic').

function [im_resized] = sp_resize(im_orig, x_new, method)

[y_or, x_or, z_or]    = size(im_orig);

y_new           = floor(y_or * x_new / x_or);

im_resized      = imresize(im_orig,[y_new x_new],method);
```

Spectral_color

```
% Ejemplo de Spectral clustering basado en
% los colores de una imagen RGB
% agrupando píxeles según el color.
% no se toma en cuenta la textura del entorno
% del píxel
% Los autovectores de la matriz de afinidad se
% calculan por medio de la aproximación
% de Nystrom.
% John Peña.  Diciembre de 2004.


clc;
clear;
close all;
tic
k               = 3;% Número de clusters deseado

% Cargando la imagen, cambiando el tamaño
% convirtiéndola en una
% matriz cuyas filas indican cada pixel y cuyas
% columnas indican los valores R G B

[I, im_orig]    = sp_open('muna.jpg', 120,'bilinear',1);
[x y z]         = size(I);
H               = reshape(double(I), x*y,z);
H               = H/255;
t1              = toc;

% Calculando distancias entre las muestras (Matriz A)
% y distancias entre las muestras y el resto de datos(Matriz B).
disp('Calculando matrices A y B...');
[ind,A, B]      = nys_AB_euc(H,2.5,30);
t2              = toc;
str             = sprintf('%f segundos.  Terminado',t2-t1);
disp(str)
disp(' ');

% Calculando los Autovectores de la matriz
% de similaridad por medio del método de Nystrom.
disp('Aproximación de Nystrom...');
[V,ss]          = nys_meth(A,B);
t3              = toc;
str             = sprintf('%f segundos.  Terminado',t3-t2);
disp(str)
[N1,N2,N3]      = size(I);
V(ind,:)        = V;
Vecs            = V(:,1:k); % escogiendo los k eigenvectores
Vnormz          = Vecs./repmat(sqrt(sum(Vecs.^2,2)),1,k);
t4              = toc;
```

```
disp(' ');

% Realizando el clustering por medio de Kmeans
% Sobre los autovectores correspondientes
disp('Ejecutando kmeans... ');
ni              = 150; %número de iteraciones para Kmeans
labels          = kmeansML(k,Vnormz','maxiter', ni);
%labels          =
kmeans(Vnormz,k,'maxiter',ni,'start','uniform','replicates', 20,
'emptyaction','singleton');
t5              = toc;
str             = sprintf('%f segundos.  Terminado',t5-t4);
disp(str)
disp(' ');

% Graficando las imágenes con los clústers corespondientes
disp('Graficando...')
imshow(I,[0 255]);

labels          = reshape(labels,[N1 N2]);

lr =[1:k]; %1 2 3 4
rl = fliplr(lr);%4 3 2 1
figure;

for g = 1:k
    mask = changem(labels,[zeros(1,k-g)], [lr(1:k-g)]);
    mask = changem(mask,[zeros(1,g-1)], [rl(1:g-1)]);
    mask = mask/max(max(mask));
    maskf(:,:,1)= mask;
    maskf(:,:,2)= mask;
    maskf(:,:,3)= mask;
    pre         =(double(I).*maskf);
    subplot(1,k,g),imshow(pre./max(max(max(pre))),[0 255]);
end

figure, imshow(labels,[1 k]);
t6              = toc;
str             = sprintf('%f segundos.  Terminado',t6-t5);
disp(str)
disp(' ');
str             = sprintf('Total: %f segundos.',t6);
disp(str)
```

spectral_int.m

```
% Ejemplo de Spectral clustering basado en la
% intensidad de los píxeles.
% Los autovectores de la matriz de afinidad se
% calculan por medio de la aproximación
% de Nystrom.  No se utiliza información de Color.
% John Peña.  Diciembre de 2004.

clc;
clear;
%close all;
tic
k               = 3; % Número de clusters deseado

% Cargando la imagen, conviertiéndola a escala
% de grises y cambiando el tamaño y graficándola.
[I, im_orig]    = sp_open('muna.jpg', 160,'bicubic',0);

I               = double(I)/max(double(I(:)));
t1              = toc;

% Calculando distancias entre las muestras (Matriz A)
% y distancias entre las muestras y el resto de datos (Matriz B).

disp('Calculando matrices A y B...');
[ind,A, B]      = nys_AB_euc(I(:),2.5,30);
t2              = toc;
str             = sprintf('%f segundos.  Terminado',t2-t1);
disp(str)
disp(' ');

% Calculando los Autovectores de la matriz
% de similaridad por medio del método de Nystrom.

disp('Aproximación de Nystrom...');
[V,ss]          = nys_meth(A,B);
V(ind,:)        = V;
Vecs            = V(:,1:k); % escogiendo los k eigenvectores
Vnormz          = Vecs./repmat(sqrt(sum(Vecs.^2,2)),1,k);
clear A B Vecs V;
t3              = toc;
str             = sprintf('%f segundos.  Terminado',t3-t2);
disp(str)
t4              = toc;
disp(' ');

% Realizando el clustering por medio de Kmeans
% Sobre los autovectores correspondientes
ni              = 150; %número de iteraciones para Kmeans
disp('Ejecutando kmeans... ');
```

```
labels          = kmeansML(k,Vnormz','maxiter',ni);
%labels          =
kmeans(Vnormz,k,'maxiter',ni,'start','uniform','replicates', 20,
'emptyaction','singleton');
clear Vnormz
t5              = toc;
str             = sprintf('%f segundos.  Terminado',t5-t4);
disp(str)
disp(' ');


% Graficando las imágenes con los clústers corespondientes
disp('Graficando...')
graf(I, im_orig, labels, k);
t6              = toc;
str             = sprintf('%f segundos.  Terminado',t6-t5);
disp(str)
disp(' ');
str             = sprintf('Total: %f segundos.',t6);
disp(str)

clear t1 t2 t3 t4 t5 t6 str ss ni k ind
```

spectral_color_3d

```
% Ejemplo de Spectral clustering basado en
% la textura de una imagen RGB
% medida por medio de un histograma tridimesnional
% calculado sobre una ventana deslizante alrededor
% de cada píxel.
% Los autovectores de la matriz de afinidad se
% calculan por medio de la aproximación
% de Nystrom.
% John Peña.  Diciembre de 2004.


tic
clc;
clear;
close all;

k               = 3; % Número de clusters deseado

% Cargando la imagen,
[I, im_orig]    = sp_openx('tex.jpg', 100,'bicubic',1);
% Calculando una matriz en la cada fila contiene
% 64 valores correspondientes a los maximos valores de  histogramas
% calculados sobre una ventana deslizante alrededor de cada pixel
% en los planos R G B respectivamente.

disp('Calculando matriz de histogramas...');

H               = hist3d(double(I)/255,7);
t1              = toc;
str             = sprintf('%f segundos.  Terminado',t1);
disp(str);
disp(' ');

% Calculando distancias entre las muestras (Matriz A)
% y distancias entre las muestras y el resto de datos.
disp('Calculando matrices A y B...');
[ind,A, B]      = nys_AB_euc(H,2.5,40);
clear H
t2              = toc;
str             = sprintf('%f segundos.  Terminado',t2-t1);
disp(str)
disp(' ');

% Calculando los Autovectores de la matriz
% de similaridad por medio del método de Nystrom.
disp('Aproximación de Nystrom...');
[V,ss]          = nys_meth(A,B);
clear A B
t3              = toc;
```

```
str             = sprintf('%f segundos.  Terminado',t3-t2);
disp(str)
[N1,N2,N3]      = size(I);
V(ind,:)        = V;
Vecs            = V(:,1:k); % escogiendo los k eigenvectores
Vnormz          = Vecs./repmat(sqrt(sum(Vecs.^2,2)),1,k);
t4              = toc;
disp(' ');
clear V Vecs
% Realizando el clustering por medio de Kmeans
% Sobre los autovectores correspondientes.
disp('Ejecutando kmeans... ');
ni              = 150;% maximo número de iteraciones
%labels          = kmeansML(k,Vnormz','maxiter', ni);
labels          =
kmeans(Vnormz,k,'maxiter',ni,'start','uniform','replicates', 30,
'emptyaction','singleton');
t5              = toc;
str             = sprintf('%f segundos.  Terminado',t5-t4);
disp(str)
disp(' ');
clear Vnormz
% Graficando las imágenes con los clústers corespondientes
disp('Graficando...')
imshow(I,[0 255]);
labels          = reshape(labels,[N1 N2]);
lr =[1:k]; %1 2 3 4
rl = fliplr(lr);%4 3 2 1
figure;
for g = 1:k
    mask = changem(labels,[zeros(1,k-g)], [lr(1:k-g)]);
    mask = changem(mask,[zeros(1,g-1)], [rl(1:g-1)]);
    mask = mask/max(max(mask));
    maskf(:,:,1)= mask;
    maskf(:,:,2)= mask;
    maskf(:,:,3)= mask;
    pre         =(double(I).*maskf);
    subplot(1,k,g),imshow(pre./max(max(max(pre))),[0 255]);
end
figure, imshow(labels,[1 k]);
t6              = toc;
str             = sprintf('%f segundos.  Terminado',t6-t5);
disp(str)
disp(' ');
str             = sprintf('Total: %f segundos.',t6);
disp(str)
clear t1 t2 t3 t4 t5 t6 str ss rl pre ni maskf ind
```

polysegment.m

```
% calcula el número inicial de grupos por medio de segmentación
% polinomial.
clc;
clear;
close all;
tic
k               = 4; % Número de clusters deseado

% Cargando la imagen, conviertiéndola a escala
% de grises y cambiando el tamaño.
[I, im_orig]    = sp_open('232038.jpg', 150,'bicubic',0);

[x, y] = size(I);
%calculando partición inicial
[rots nvecto] = spoly(double(I));

nvecto2 = reshape(nvecto, x,y);
figure, imshow(nvecto2/max(nvecto2(:)));
figure, imshow(im_orig);
rots
```

spectral_colorhist_OE.m

```
% Ejemplo de Spectral clustering basado en
% la textua de una imagen RGB y en contornos
% Los autovectores de la matriz de afinidad se
% calculan por medio de la aproximación
% de Nystrom.
% John Peña.  Diciembre de 2004.

tic
clc;
clear;
close all;

k               = 5; % Número de clusters deseado

% Cargando la imagen,
[I, im_orig]    = sp_open('232038.jpg', 130,'bicubic',1);
[Ig, im_or_g]   = sp_open('232038.jpg', 130,'bicubic',0);

[VecN emag] = sp_OE_Nvecs(3, Ig, 0.15);
%VecN = [];
disp('listo contornos')
cont = toc
% Calculando una matriz en la cada fila contiene
% 3 valores correspondientes a los maximos valores de  histogramas
% calculados sobre una ventana deslizante alrededor de cada pixel
% en los planos R G B respectivamente.

disp('Calculando matriz de histogramas...');

H               = double(sp_hvec_1D_color_xl(I,9));
t1              = toc;
str             = sprintf('%f segundos.  Terminado',t1-cont);
disp(str);
disp(' ');

% Calculando distancias entre las muestras (Matriz A)
% y distancias entre las muestras y el resto de datos.
disp('Calculando matrices A y B...');
[ind,A, B]      = nys_AB_euc(H,3,40);
t2              = toc;
str             = sprintf('%f segundos.  Terminado',t2-t1);
clear H;
disp(str)
disp(' ');

% Calculando los Autovectores de la matriz
% de similaridad por medio del método de Nystrom.
disp('Aproximación de Nystrom...');
```

```
[V,ss]          = nys_meth(A,B);
t3              = toc;
str             = sprintf('%f segundos.  Terminado',t3-t2);
disp(str)
[N1,N2,N3]      = size(I);
V(ind,:)        = V;
Vecs            = V(:,1:k); % escogiendo los k eigenvectores
Vnormz          = Vecs./repmat(sqrt(sum(Vecs.^2,2)),1,k);
Vf              = [Vnormz,VecN];
clear A B;
t4              = toc;
disp(' ');
clear Vnormz Vecs VecN
% Realizando el clustering por medio de Kmeans
% Sobre los autovectores correspondientes.
disp('Ejecutando kmeans... ');
ni              = 150;% maximo número de iteraciones
labels          = kmeansML(k,Vf','maxiter', ni);
%labels          =
kmeans(Vf,k,'maxiter',ni,'start','uniform','replicates', 10,
'emptyaction','singleton');
t5              = toc;
str             = sprintf('%f segundos.  Terminado',t5-t4);
disp(str)
disp(' ');
clear V Vecs ;
clear Vf
% Graficando las imágenes con los clústers corespondientes
disp('Graficando...')
imshow(I,[0 1]);
labels          = reshape(labels,[N1 N2]);
lr =[1:k]; %1 2 3 4
rl = fliplr(lr);%4 3 2 1
figure;
for g = 1:k
    mask = changem(labels,[zeros(1,k-g)], [lr(1:k-g)]);
    mask = changem(mask,[zeros(1,g-1)], [rl(1:g-1)]);
    mask = mask/max(max(mask));
    maskf(:,:,1)= mask;
    maskf(:,:,2)= mask;
    maskf(:,:,3)= mask;
    pre         =(double(I).*maskf);
    subplot(1,k,g),imshow(pre./max(max(max(pre))),[0 255]);
end
figure, imshow(labels,[1 k]);
t6              = toc;
str             = sprintf('%f segundos.  Terminado',t6-t5);
disp(str)
disp(' ');
str             = sprintf('Total: %f segundos.',t6);
disp(str)
clear mask maskf pre t1 t2 t3 t4 t5 t6 N1 N2 N3 ss I k lr ni rl ind
```

8 BIBLIOGRAFÍA

[1] Boult, T.E. and Brown, L.G. *Factorization-based segmentation of motions*. En *Proc. of the* IEEE Workshop on Motion Understanding, páginas 179–186, 1991.

[2] Boyer, Kim and Sarkar, Sudeep. *Quantitative measures of change based on feature organization: eigenvalues and eigenvectors*. Computer vision and image understanding. Vol. 71, No. 1, July, pp. 110–136, 1998.

[3] Boyer, Kim and Sarkar, Sudeep. *Perceptual Organization in Computer Vision: Status, Challenges, and Potential*. Computer Vision and Image Understanding. Vol. 76, No. 1, October, pp. 1–5, 1999.

[4] Chastel, S. *Color Image Segmentation - Histogram*. Institut für Computer Visualistik Universität Koblenz-Landau Universitätsstraße. http://www.uni-Koblenz.de/~chastel. 2003.

[5] Chung, Fan R. K. *Spectral Graph Theory*. American Mathematical Society, 1997.

[6] Costeira, J. and Kanade, T. *A multibody factorization method for independently moving objects*. International Journal of Computer Vision, 29(3):159–179, 1998.

[7] Diestel, Reinhard. *Graph Theory*, Electronic Edition 2000, Springer-Verlag New York 1997, 2000.

[8] Forsyth, D. y Ponce, J. *Computer Vision: A Modern Approach*. Prentice Hall, 2002.

[9] Fowlkes, C., Belongie, S., Chung F.R. y Malik, J. *Spectral grouping using the Nyström method*. IEEE transactions on pattern analysis and machine intelligence, vol. 26, no. 2, february 2004.

[10] Fowlkes, C., Belongie, S., Chung F.R. y Malik, J. *Spectral grouping with indefinite kernels using the Nyström method*. Procedings European Conference Computer Vision, 2002.

[11] Freeman, W. T. y Adelson E. H. *The desing and use of steerable filters*. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 13, No. 9, páginas 891-906. 1991.

[12] Hogben, Leslie. *Spectral graph theory and the inverse eigenvalue problem of a graph*. Department of Mathematics, Iowa State University, AMS subject classifications. 05C50, 15A18. 1996.

[13] Kanatani, K. *Motion segmentation by subspace separation and model selection*. IEEE International Conference on Computer Vision, volume 2, páginas 586–591, 2001.

[14] Kleitman, Daniel. *Some graph theory*. Massachusetts Institute of Technology Lecture Notes. http://books.pdox.net.

[15] Malik, J., Belongie, S., Leung, T. y Shi, J. Contour and texture analysis for image segmentation. International Journal of Computer Vision 43(1), páginas 7-27, 2001.

[16] Malik, J., Belongie, S., Leung, T. Y Shi, J. *Textons, contours and regions: cue integration in image segmentation*. IEEE International Conference on Computer Vision, Corfu, Greece, September 1999.

[17] Martin, D.,Fowlkes, Charless y Malik, J. *Learning to detect natural image boundaries using local brightness, color and texture cues*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 1, 2004.

[18] Martin, D., Fowlkes, C., Malik, J. y Tal, D. *A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics*. IEEE International Conference on Computer Vision, 2001.

[19] Morrune y Burr. Feature detection in human vision: a phase dependent energy model. Proc. R. Soc. Lond. B, 235, 1998.

[20] Morrone y Owens. Feature detection from local energy. Pattern recognition letters, 6, 1987.

[21] Perona, P. y Freeman, W. *A factorization approach to grouping*. *European Conference on Computer Vision*, páginas 655–670, 1998.

[22] Scott, G. and Longuet-Higgins, H. *Feature grouping by relocalisation of eigenvectors of the proximity matrix*. En British Conference on Machine Vision, páginas 731–737, 1990.

[23] Shi, J. and Malik, J. *Normalized cuts and image segmentation*. IEEE Trans. on Pattern Analysis and Machine Intelligence, 22(8):888–905, 2000.

[24] Stan, Daniela. *Image feature extraction*. Visual computing workshop: image processing DePaul University, 2004.

[25] Su, Yeping and Song, Xiaodan. *Normalized Cuts and Image Segmentation*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000. p.3.

[26] Vidal, E. *Generalized Principal Component Analysis (GPCA): an Algebraic Geometric Approach to Subspace Clustering and Motion Segmentation*. A dissertation submitted in partial satisfaction of the requirements for the degree of Doctor of Philosophy in Engineering – Electrical Engineering and Computer sciences in the graduate division of the University of California at Berkeley. Cap. 2.

[27] Weiss, Y., Ng, A. Y. y Jordan, M.I. *On spectral clustering: analysis and an algorithm*. Advances in Neural Information Processing Systems 14. MIT Press, 2002.

[28] Weiss, Y. Segmentation *using eigenvectors: a unifying view*. In IEEE International Conference on Computer Vision, páginas 975–982, 1999.

[29] Weisstein, Eric W. "Laplacian Matrix." From MathWorld--A Wolfram Web Resource. http://mathworld.wolfram.com/LaplacianMatrix.html

[30] Weisstein, Eric W. *"Directed Graph."* From MathWorld--A Wolfram Web Resource. http://mathworld.wolfram.com/DirectedGraph.html.

[31] Weisstein, Eric W. *"Graph."* From MathWorld--A Wolfram Web Resource. http://mathworld.wolfram.com/Graph.html.

[32] Weisstein, Eric W. *"Graph eigenvalue."* From MathWorld--A Wolfram Web Resource. http://mathworld.wolfram.com/GraphEigenvalue.html.

[33] Weisstein, Eric W. *"Weighted Graph."* From MathWorld--A Wolfram Web Resource. http://mathworld.wolfram.com/DirectedGraph.html.

[34] Yu, Stella X. *Computational models of perceptual Organization*. Dissertation submitted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Robotics at Carnegie Mellon University Pittsburgh, Pennsylvania, 2003.