Universidad del Cauca
Ingeniería en Automática Industrial
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Electrónica, Instrumentación y Control

Segmentación de imágenes mediante un algoritmo de agrupamiento espectral

Monografía presentada como requisito parcial para optar al título de Ingeniero en Automática Industrial

PABLO ANDRÉS LIZARAZO CHILAMÁ
JUAN PABLO RODRÍGUEZ FERNÁNDEZ

Director: Mg. Elena Muñoz España

Cauca, 2017

Universidad del Cauca

Universidad del Cauca
Ingeniería en Automática Industrial
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Electrónica, Instrumentación y Control

Segmentación de imágenes mediante un algoritmo de agrupamiento espectral

Monografía presentada como requisito parcial para optar al título de Ingeniero en Automática Industrial

PABLO ANDRÉS LIZARAZO CHILAMÁ
JUAN PABLO RODRÍGUEZ FERNÁNDEZ

Director: Mg. Elena Muñoz España

Nota: .....
.....

Aprobado por: Fecha: .....

.....	.....	.....
Mg. Elena Muñoz España	Firma Jurado	Firma Jurado

Cauca, 2017

Resumen

En este proyecto se propone un algoritmo de segmentación de imágenes basado en agrupamiento espectral, SFSC (*Superpixels, fuzzy and Spectral Clustering*), el cual para imágenes de tamaño considerable obtiene una segmentación en menor tiempo que un algoritmo clásico de agrupamiento espectral, además se realiza un análisis con respecto a otras métricas y algoritmos que presentan semejanza.

La codificación del algoritmo se llevó a cabo en C++. Para realizar las operaciones matriciales y dar solución al problema de valores y vectores propios se utilizó la librería *eigen*, para mejorar el desempeño de *eigen* se incluyeron librerías extras como: *Lapack*, *OpenMP*, *Blas*, *Pthread* y principalmente la librería intel® *Math Kernel Library* (MKL). Se implementa una interfaz gráfica de usuario (GUI) en el framework multi plataforma Qt y para el procesamiento digital de la imágenes se utilizó OpenCV.

El algoritmo presenta tres pilares para su desarrollo: pre-segmentación o diezmando en súper píxeles con el algoritmo *Simple Linear Iterative Clustering* (SLIC), construcción de la matriz de Similaridad con una medida difusa basada en el clasificador *Fuzzy C-Means* (FCM) y finalmente mediante agrupamiento espectral se determina la segmentación final. Con agrupamiento espectral se realiza la construcción de la matriz Laplaciana normalizada y se determinan los k vectores propios (espectro). La matriz Laplaciana normalizada como la medida difusa da un buen resultado para determinar el mejor punto corte.

Al comparar el algoritmo desarrollado con otros métodos de segmentación, se consiguieron buenos resultados en la métrica de complejidad temporal respecto al algoritmo clásico, además, se obtienen buenos resultados para las diferentes métricas de validación, internas como externas.

Palabras clave: *Agrupamiento Espectral, Valores y Vectores Propios, Súper Píxeles, Fuzzy C-Means, K-Means, Matriz Similaridad, Matriz Laplaciana.*

El presente trabajo va dirigido como una expresión de gratitud a las personas que estuvieron presentes con su voz de aliento, y en especial agradecemos a nuestros padres por su infinito amor y apoyo incondicional.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Motivación del proyecto y Estado del Arte.

1.1. Motivación

La visión por computador es una rama de la inteligencia artificial ampliamente utilizada en la actualidad, sus principales aplicaciones son: control de calidad, monitoreo, análisis de imágenes, entre otras. Uno de los pilares de la visión por computador se centra en la segmentación de imágenes; ya que de esta etapa depende una interpretación acertada de la realidad. La tarea de segmentación consiste en dividir la imagen en una serie de regiones que no se sobrepongan y que compartan características homogéneas como: intensidad, color, textura, etc [1–4]. Los algoritmos clásicos de agrupamiento espectral han sido ampliamente utilizados en la segmentación de imágenes desde hace algunas décadas; ya que han demostrado buenos resultados y son de fácil implementación. A pesar de ello, la mayoría de estos algoritmos tienen que lidiar con un elevado costo computacional y tiempo de procesamiento [5].

El aumento en el tiempo de procesamiento y costo computacional se debe principalmente a la construcción de la matriz de similaridad. Dicha matriz está conformada por todas las posibles relaciones entre pares de puntos que forman la imagen, dejando como resultado un cálculo complejo de datos y un excesivo uso de memoria [6].

Actualmente es muy común trabajar con imágenes en alta definición (HD), en general estas imágenes tienen un tamaño de 1920x1080 píxeles, imágenes con resoluciones más altas son cada vez más populares; este elevado número de datos a procesar aumenta en gran medida las desventajas de los algoritmos de agrupamiento espectral clásicos [7].

1.2. Estado del Arte

En las últimas décadas, diferentes algoritmos se han aplicado a la segmentación de imágenes [6, 8, 10–12]. Según [13] en general estos algoritmos pueden clasificarse en cuatro categorías: método de umbral, detección de bordes, métodos basados en la región y los de agrupamiento espectral; donde estos últimos se han utilizado con éxito en el campo del reconocimiento de patrones y la visión artificial [14, 15].

El principio del agrupamiento espectral se basa en la teoría de grafos y consta de las siguientes tres etapas: pre-procesamiento, descomposición y finalmente la agrupación. En la etapa de tratamiento previo, a partir de los datos obtenidos de la imagen se construye la gráfica de similitud y su matriz de similaridad. En la segunda etapa, mediante la obtención de los autovectores de la matriz de similaridad, se cambia la representación de los datos a un espacio donde es más sencillo hacer el agrupamiento [16]. Finalmente se forman conjuntos no vacíos tal que la similaridad entre los datos del mismo grupo sean altas, mientras que la de los datos entre diferentes grupos sean bajas [17]. *K-Clustering*, ISODATA, *K-Means* y HCAs, son algunos algoritmos convencionales basados en agrupamiento espectral, enfocados en la segmentación de imágenes [18]. *Fuzzy C-Means* (FCM) es otro tipo de algoritmo de agrupamiento fundamentado en la lógica difusa donde su principal característica es que un píxel puede pertenecer a varios grupos a la vez, pero con diferente grado de pertenencia [4, 9]. Sin embargo, es un hecho ampliamente reconocido que los métodos de agrupación mencionados anteriormente tienen sus propias limitaciones [10].

La correcta segmentación de imágenes se ve afectada por presencia de ruido y valores atípicos [19]. En [14] se propone un algoritmo basado en modelos de mezclas gaussianas. En primer lugar, se modela la distribución conjunta de las características de color y posición con una mezcla gaussiana; a fin de encontrar una aproximación o estimación de la imagen original. Las medidas de similaridad se realizan entre los componentes de Gauss. Finalmente, los componentes de Gauss se fusionan sobre la matriz de similaridad mediante agrupamiento espectral. Las mejores características del algoritmo son la robustez frente al ruido, baja sensibilidad a la inicialización, extracción eficaz de la característica espectral y reducción del tamaño de la matriz de similaridad. Aunque, cuanto mayor sea el número de componentes gaussianos utilizados para la recreación de la imagen, la carga computacional y el sobre ajuste de datos también será más alto.

El diagrama de Voronoi soportado en los polígonos de Thiessen, tiene como objetivo, el

teselado de una imagen, esto quiere decir dividir dicha imagen en regiones con forma de polígonos irregulares, donde estos polígonos están formados por rectas, semirrectas o segmentos de rectas, el diagrama de Voronoi debe cumplir dos propiedades: que no queden espacios en el plano sin cubrir y que los polígonos no se sobrepongan. Para esto se ubican k puntos en la imagen que son llamados los centroides de cada región. Cada píxel de la imagen se vincula al centroide con el que comparta más similitud [26, 28]. Bajo esta técnica los algoritmos FCM funcionan bien siempre que la imagen está formada por regiones homogéneas, pese a que, estos algoritmos sólo tienen en cuenta la información de intensidad de los píxeles, pero no la información espacial, la segmentación de imágenes también se ve afectada por el ruido [26]. En [26] proponen el algoritmo FEWCVT (*fuzzy edge-weighted centroidal Voronoi tessellations*), el cual genera una función de similitud difusa combinando la información de intensidad de la imagen con la información de los límites de las agrupaciones. Consiguiendo así bordes más suaves y mayor tolerancia al ruido.

Otra propuesta realizada para el problema del ruido se describe en [19]. Se da solución aplicando un algoritmo difuso en la obtención de la matriz de similaridad. Primero, se construye una imagen en grises como la suma ponderada no local de la imagen original, seguidamente con el algoritmo difuso se determinan las medidas de similaridad haciendo uso de los valores en grises (Información Espacial). Con la matriz de similaridad construida se aplica agrupamiento espectral. Para finalizar los píxeles son asignados al grupo correspondiente con mayor similaridad.

La mayoría de los algoritmos de agrupamiento espectral utilizan la función gaussiana como medida de similitud [27], sin embargo esta no puede reflejar plenamente la compleja distribución espacial del conjunto de datos, y es indeseable cuando los grupos de datos se desarrollan sobre una estructura compleja [15, 20, 21]. El problema que se aborda en [22] es la construcción de la matriz de similaridad, se requiere encontrar una medida de similaridad más robusta entre pares de datos en contraste con la medida producida por la función núcleo gaussiana. Se propone un método sin supervisión llamado AFSSC (*Axiomatic Fuzzy Set-based Spectral Clustering*) que aprovecha las características discriminantes. Este método es capaz de capturar y combinar información de similaridad distribuida en subespacios, llevando a revelar con mayor precisión la distribución de los datos. El método se desempeñó satisfactoriamente en grupos de datos que contenían una gran cantidad de características redundantes y/o componentes con ruido.

Una medida de similitud difusa, aplicada a la segmentación de imágenes con textura, se presenta en [15]. Inicialmente, se genera la matriz U de agrupamiento difuso, en la cual

las filas representan el número de particiones, y las columnas el número total de píxeles. Así entonces cada elemento de la matriz contiene el grado de pertenecía del píxel j con respecto a la partición i. Los píxeles están caracterizados por un total de 10 atributos, donde los ocho primeros dan información acerca de la textura y los dos últimos su ubicación espacial, estos atributos son el criterio para asignar el grado de pertenencia hacia un grupo determinado. Como segundo paso, se construye la matriz de similaridad a partir de la matriz U, puede verse como dos píxeles son muy similares si comparten los mismos grados de pertenencia, por el contrario dos píxeles diferentes no tendrán concordancia en sus grados de pertenencia, por último se aplican los pasos clásicos de agrupación espectral para segmentar la imagen.

Finalmente, ya que la matriz de similaridad de una imagen representa las afinidades entre todos sus píxeles, su tamaño suele ser demasiado grande, esto hace que sea difícil computar dicha matriz [13, 22, 23]. Aplicaciones basadas en súper píxeles solucionan este problema. Para el desarrollo de súper píxeles se han propuesto varios algoritmos basados en grafos y otros, basados en el crecimiento del gradiente [24]. Uno de los métodos más prometedores para la generación de súper píxeles con bajo costo computacional y gran velocidad de procesamiento se presenta en [13]. Se propone realizar la segmentación en dos etapas. Primero, se pre-segmenta la imagen por el método de SLIC (*Simple linear Iterative Clustering*) modificado, donde se extraen las regiones "globales" que contengan características similares [23]. Segundo, se realiza el agrupamiento de las regiones pre-segmentadas en lugar de los píxeles, haciendo uso del método de Nyström. Este método permite extrapolar la solución de agrupamiento utilizando un pequeño número de muestras.

Si la imagen contiene muchos píxeles exige más recursos computacionales, y mayor tiempo de procesamiento en la solución de la matriz de similaridad. Para resolver el enorme cálculo de la matriz, en [5] se introduce un algoritmo de súper píxeles llamado SCS (*Superpixel Spectral Clustering*), el cual permite dividir la imagen en un conjunto conectado de regiones homogéneas [25]; con la finalidad de obtener la matriz de similaridad de las regiones resultantes. Se muestra que SCS es extremadamente estable y apenas se ve afectado por el número de vecinos más cercanos, además, se puede lograr una precisión comparable y realiza significativamente mejor el proceso de agrupamiento que la mayoría de los algoritmos clásicos.

En resumen a lo expuesto anteriormente se concluye que: los trabajos realizados en [14, 19, 26], están enfocados en dar solución al problema de ruido presente en algunas imágenes. Del mismo modo los autores de [15, 22, 27] han centrado sus esfuerzos en proponer funciones de similitud difusas, para conseguir agrupaciones más óptimas. Por último en [5, 13] se tratan técnicas para reducir los datos a ser procesados.

1.3. Objetivos

El objetivo principal de este proyecto es proponer un algoritmo de segmentación de imágenes bajo la técnica del agrupamiento espectral, de forma que minimice los datos a procesar y disminuya el tiempo de ejecución.

A continuación, se presentan los tres objetivos específicos que se pretenden alcanzar con este trabajo.

1. Diseñar un algoritmo de agrupamiento espectral para la segmentación de imágenes, incluyendo en la etapa de pre-procesamiento técnicas de agrupamiento difuso y de súper píxeles.

2. Implementar el algoritmo de segmentación para obtener el objeto de interés separado del resto de la imagen.

3. Evaluar el desempeño del algoritmo implementado, comparándolo con una técnica clásica de agrupamiento espectral.

1.4. Resumen de la propuesta

En el presente trabajo se propone un algoritmo para la segmentación de imágenes basado en técnicas de agrupamiento espectral, donde su principal objetivo es reducir el tiempo de ejecución y la memoria empleada para tal fin, en comparación con un algoritmo clásico de agrupamiento espectral.

El principal foco consiste en aliviar la construcción de la matriz de similaridad reduciendo los datos que la componen, para esto se hace uso de técnicas de súper píxeles, diezmando así la imagen sin afectar sus propiedades.

Aparte de estar constituida por menos datos, la construcción de esta matriz se basa en una función de similitud difusa, para lograr un criterio menos rígido a la hora de clasificar los píxeles, proporcionando una mejor segmentación, además, de brindarle al algoritmo una mayor robustez frente a datos atípicos o presencia de ruido en la imagen.

El algoritmo ha sido implementado en C++ y cuenta con una interfaz gráfica desarrollada en QT Creator, por medio de la cual el usuario puede seleccionar la imagen a segmentar y definir los parámetros de segmentación.

Finalmente, se evalúa el desempeño del algoritmo basándose en algunas métricas existentes, tales como: tiempo, compacidad, dispersión, complejidad espacial, con el objetivo de conseguir resultados que puedan ser comparados con un algoritmo clásico de agrupamiento y bajo estos criterios definir si el algoritmo propuesto en este trabajo, supera en algún aspecto al algoritmo de referencia.

1.5. Estructura del documento

Este trabajo está dividido en seis capítulos (incluido el presente), que siguen una traza lógica y que van desde la base teórica hasta la implementación y validación del algoritmo propuesto. Así como las conclusiones y trabajos futuros. En el **Capítulo II**, se trata los aspectos claves para que el lector pueda alcanzar una comprensión detallada del proceso que se lleva a cabo en cada uno de los algoritmos que componen esta propuesta. Describe las raíces teóricas de los algoritmos de agrupamiento espectral, súper píxeles y agrupamiento difuso. En el **Capítulo III**, se detalla la propuesta de este trabajo de grado. De esta manera, se aborda el proceso de diseño del algoritmo para la segmentación de imágenes, súper píxeles, *fuzzy and spectral clustering* (SFSC) profundizando en todos los detalles acerca de la implementación de las técnicas que comprenden este algoritmo, también, se proporciona el pseudocódigo del algoritmo desarrollado.

En el **Capítulo IV**, se presentan algunas técnicas de evaluación para los algoritmos de agrupación, su definición y las métricas que utilizan. En el **Capítulo V**, se lleva a cabo la validación experimental de la propuesta. A través de un conjunto heterogéneo de imágenes, se evalúa la calidad de los resultados obtenidos, comparando el algoritmo desarrollado con un algoritmo clásico de agrupamiento espectral.

En el **Capítulo VI**, se presentan las conclusiones finales a las que se ha llegado después del estudio y aplicación del algoritmo implementado. También se plantean las líneas de investigación del proyecto y algunas propuestas para trabajos futuros.

Capítulo 2

Marco conceptual.

Este capítulo presenta la base teórica necesaria para un buen desarrollo de los capítulos posteriores. En la sección 2.1 y 2.2 se definen los elementos básicos de la teoría de grafos y la teoría espectral de grafos respectivamente, que son la base de los algoritmos de agrupamiento espectral clásicos. En la sección 2.3 se define el agrupamiento difuso, y en 2.4 se introduce el concepto de súper píxel y se explica cómo reduce los datos de una imagen guardando la naturaleza de la misma.

2.1. Teoría de Grafos

Atribuida a Leonard Euler a principios de 1730, la teoría de grafos es una rama de las matemáticas, compuesta por diferentes áreas como: estadística, combinatoria, álgebra, geometría entre otras, donde sus principales aplicaciones son la optimización de procesos y recorridos. En la actualidad es de mucho interés en las ciencias de la computación, la informática y las telecomunicaciones.

2.1.1. Definición de grafo

Un grafo es un par de conjuntos $G = (V, E)$ que satisfacen $E \subseteq [V]^2$, donde los elementos de V son los vértices (nodos o puntos) y los elementos de E son llamados aristas (bordes o arcos). La forma más común de representar un grafo es dibujando un punto o círculo por cada vértice; y la unión de estos puntos se hace a través de líneas que representan

las aristas. Un grafo completo es aquel donde a cada vértice inciden $V-1$ aristas. El conjunto de vértices de un grafo G se denota como $V(G)$ y su conjunto de bordes como $E(G)$ [28–31]. El orden de un grafo G equivale a su número de vértices [29] como se muestra en la Figura 2.1.

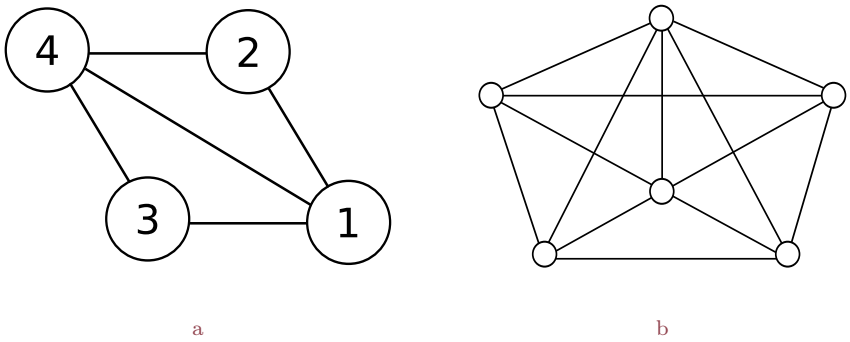


Figura 2.1: *Representación de un grafo. a: Grafo compuesto por 4 vértices y 5 aristas. b: Grafo completo. (Adaptada de [29])*

De la Figura 2.1a, se obtiene:

Orden del grafo: $G = 4$.

Conjunto de vértices: $V(G) = [1, 2, 3, 4]$.

Conjunto de bordes: $E(G) = [(1,2)(1,3)(1,4)(2,4)(3,4)]$.

2.1.2. Grafo ponderado

Un grafo con pesos, o grafo ponderado, es un grafo en el que a cada arista se le asigna un valor real no negativo o peso. Se denotará por w_{ij} el peso del lado que une el vértice v_i con el vértice v_j. Si $w_{ij} = w_{ji}$, se hablará entonces de un grafo no dirigido. A un grafo sin pesos se le conoce como grafo no ponderado. El grado de un vértice v_i es la suma de los pesos de todos los lados incidentes en v_i, el cual, es denotado por $d(v_i)$. Un vértice de grado cero se dice que está aislado [30]. En la Figura 2.2 se visualiza la estructura de un grafo ponderado.

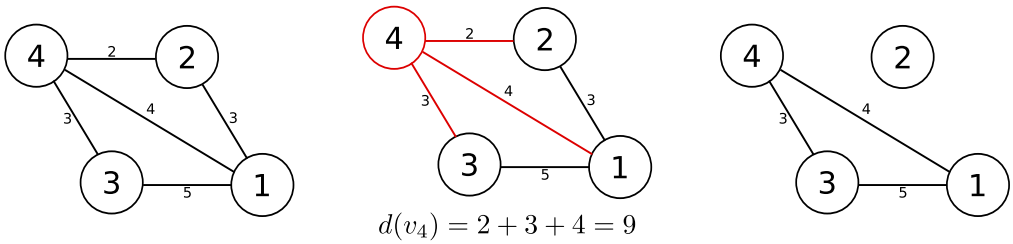


Figura 2.2: *Grafos. a: Grafo ponderado de 4 vértices. b: Grado del vértice* v_4 $d(v_4) = 9$. *c: Vértice aislado. (Realizada por los autores)*

2.1.3. Matriz de adyacencia

Dos vértices son adyacentes o vecinos si existe una arista que los une, se denota esta relación como: $v_i \sim v_j$. Sea $G = (V, E)$ un grafo ponderado, donde $V = \{v_1, v_2, \ldots, v_n\}$. La matriz de adyacencia $A(G)$ de G es la matriz de orden $n \times n$ donde n es el orden de G, esta matriz se denota como [29, 30].

$$A(G) = \begin{cases} W_{ij} & Si\ v_i \sim v_j \\ 0 & Otro\ caso \end{cases} \tag{2.1}$$

En el caso de ser G un grafo no ponderado, la relación $v_i \sim v_j$ es igual a 1, como se muestra en la Figura 2.3.

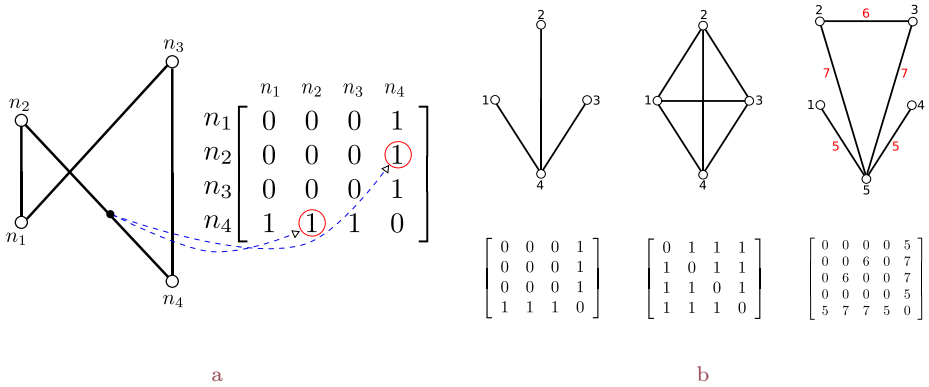


Figura 2.3: *Construcción de la matriz de adyacencia. a: Construcción. b: Grafos con su matriz de adyacencia. (Adaptada de [29])*

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 (i, j) de la matriz es igual a 1, de lo contrario es igual a cero. En grafos ponderados, si los vértices (v_i, v_j) son adyacentes el elemento (i, j) de la matriz, toma el valor de la arista correspondiente según la función de peso (w) aplicada entre estas.

2.1.4. Matriz de similaridad

La matriz de similaridad es un concepto derivado de la teoría de grafos, y representa las relaciones de similaridad entre todos los píxeles de una imagen. Una matriz de similaridad ideal S asociada a una imagen P , 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 [44].

2.1.4.1. Función de afinidad

Con frecuencia en la literatura la función de peso $W(i, j)$ suele llamarse medida de afinidad, esta medida puede basarse; en la distancia espacial, textura, intensidad, color, etc [47], y para poder ser utilizada en la construcción de la matriz de similaridad debe cumplir las siguientes condiciones:

- $0 \leq S(x_i, x_j) \leq 1$
- $S(x_i, x_i) = 1$
- $S(x_i, x_j) = S(x_j, x_i)$

Donde x_i, x_j son los vectores que caracterizan a los píxeles i, j respectivamente. Una expresión apropiada para garantizar estas condiciones tiene la forma:

$$W(i, j) = exp^{-\left\|x_i - x_j\right\|^2 / 2\sigma^2}$$

Donde σ es el parámetro que determina qué tan rápido decaen las afinidades a medida que aumenta la distancia entre las características de los píxeles, generalmente se calcula experimentalmente. $\|x_i - x_j\|$ es la distancia entre los vectores que caracterizan a los píxeles i y j [44, 47].

2.1.5. Punto de corte

Una arista $\{i, j\} \in E(G)$ ligada a un grafo G es llamada puente o arista de corte si al ser eliminada consigue dividir el grafo en dos partes [32]. Como se muestra en la Figura 2.4.

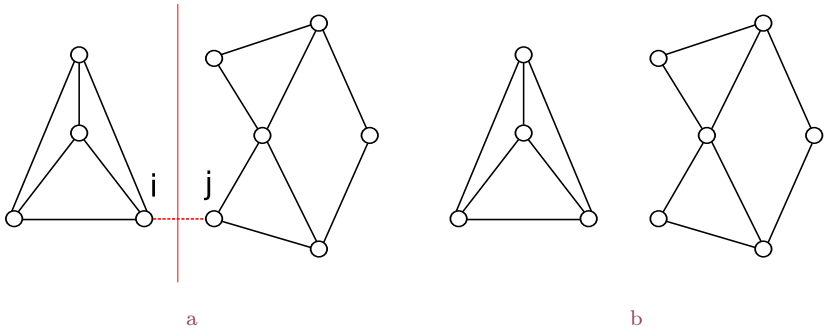


Figura 2.4: *Partición de un grafo. a: Identificación de la arista del corte* $\{i, j\}$. *b: Partición del grafo en dos regiones A y B respectivamente. (Adaptada de [32])*

De manera general un grafo $G = (V, E)$ puede dividirse en dos partes disjuntas (A, B), con $A \bigcup B = V$ y $A \bigcap B = 0$ eliminando las aristas que representan baja relación entre los vértices [28, 32, 34]. Esto se consigue minimizando el valor de corte $Cut(A, B)$. Definido como :

$$Cut(A, B) = \sum_{i \in A, j \in B} W(i, j) \tag{2.2}$$

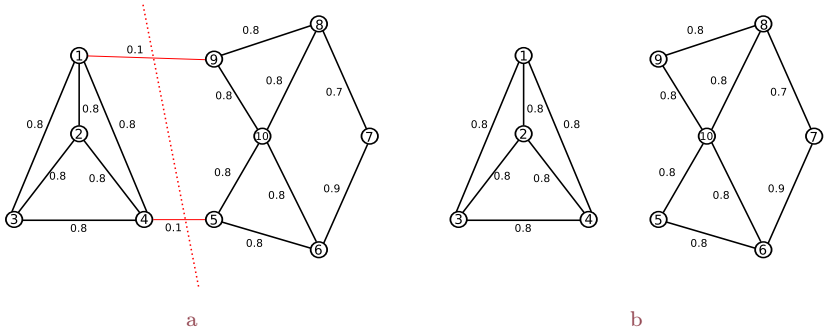


Figura 2.5: *Punto de corte de un grafo. a: Selección punto de corte. b: División en dos grupos* (A, B). *(Adaptada de [32])*

En la Figura 2.5a el mínimo valor de corte posible para dividir el grafo es igual a la suma de los valores de las aristas atravesadas por la línea de corte (línea punteada). El valor de corte resultante es:

$$Cut(A,B) = \sum_{i \in A, j \in B} W(i,j) = W(1,9) + W(4,5) = 0.2$$

La teoría de grafos es bastante amplia debida a su gran cantidad de aplicaciones. En este trabajo se introducen sólo los conceptos necesarios para un buen desarrollo del mismo; pero el lector podrá profundizar sobre este tema en [29].

2.2. Teoría espectral de Grafos

Atribuida a Donald y Hoffman a principios de los años 1970, 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 de la teoría espectral de grafos es deducir las propiedades y estructura de un grafo a partir de su espectro y vectores propios [28, 35, 46].

2.2.1. Matriz Laplaciana

En la literatura se encuentra una gran variedad de definiciones para la matriz Laplaciana (simple, normalizada, generalizada), cada definición se justifica por propiedades especiales dentro de un contexto dado [28]. Todas estas definiciones comparten la matriz de grado que consiste en una matriz diagonal tal que:

$$D(i,i) = \sum_{j}^{n} a_{ij} \tag{2.3}$$

En otras palabras D_{ii} sería una matriz diagonal en la que el elemento (i,i) sería igual a la suma de los elementos de la fila i de la matriz de similaridad $S(G)$ o de la matriz de adyacencia $A(G)$ [28, 31, 33–35]. Por lo tanto, la matriz Laplaciana es:

$$\begin{array}{ll} \textit{Laplaciana Simple} & L = D - A \\ \textit{Laplaciana Normalizada} & L_n = D^{-1/2} L D^{-1/2} \end{array} \tag{2.4}$$

A continuación se aprecia la construcción de la matriz Laplaciana simple y normalizada a partir del grafo ponderado G, de la Figura 2.6.

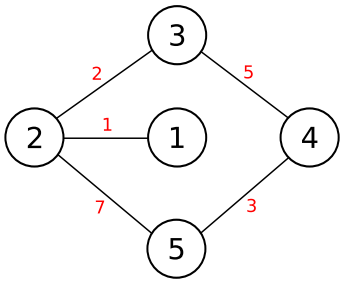


Figura 2.6: *Grafo ponderado G. (Realizada por los autores)*

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 2 & 0 & 7 \\ 0 & 2 & 0 & 5 & 0 \\ 0 & 0 & 5 & 0 & 3 \\ 0 & 7 & 0 & 3 & 0 \end{pmatrix} \quad D_{ii}(A) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 7 & 0 & 0 \\ 0 & 0 & 0 & 8 & 0 \\ 0 & 0 & 0 & 0 & 10 \end{pmatrix} \quad D_{ii}(A)^{-1/2} = \begin{pmatrix} \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{20} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{14} & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{16} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{20} \end{pmatrix}$$

$$L = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 10 & -2 & 0 & -7 \\ 0 & -2 & 7 & -5 & 0 \\ 0 & 0 & -5 & 8 & -3 \\ 0 & -7 & 0 & -3 & 10 \end{pmatrix} \quad L_n = \begin{pmatrix} \frac{1}{4} & -\frac{1}{40} & 0 & 0 & 0 \\ -\frac{1}{40} & \frac{1}{40} & -\frac{1}{140} & 0 & -\frac{7}{400} \\ 0 & -\frac{1}{140} & \frac{1}{28} & -\frac{5}{224} & 0 \\ 0 & 0 & -\frac{5}{224} & \frac{1}{32} & -\frac{3}{320} \\ 0 & -\frac{7}{400} & 0 & -\frac{3}{320} & \frac{1}{40} \end{pmatrix}$$

2.2.2. Valores y Vectores propios

Sea L una matriz cuadrada, se dice que λ es un valor propio (espectro) de L, si cumple que:

$$det[\lambda I - L] = 0 \tag{2.5}$$

Siendo I la matriz identidad y L la matriz Laplaciana. Debido a que la matriz de adyacencia

y la Laplaciana son reales y simétricas; sus valores propios también son números reales.

Si X es un vector de orden n donde $X = \{x_1, x_2, \ldots, x_n\}$, se dirá que es un vector propio de L si se cumple que [33]:

$$LX = \lambda X \tag{2.6}$$

En la sección siguiente se demostrará cómo los valores y vectores propios de la matriz Laplaciana normalizada proporcionan un criterio bastante sólido para la segmentación de una imagen.

2.2.3. Agrupamiento espectral

La relación global entre las características de una imagen puede ser representada de manera efectiva por un grafo completo no dirigido también conocido como grafo de similitud, cuyos vértices hacen referencia a las características asociadas a cada píxel y los arcos que unen los vértices indican la similaridad entre estos. Siendo así, el hecho de lograr una partición coherente de un grafo implica también lograr una partición coherente en la imagen [28].

La partición se logra minimizando el valor de corte (ecuación 2.2), consiguiendo eliminar los bordes que representan una relación débil entre grupos, sin embargo, este procedimiento puede conducir a resultados erróneos ya que este criterio favorece la formación de pequeños subconjuntos de vértices aislados [36]. En la Figura 2.7 se ilustra la formación de vértices aislados.

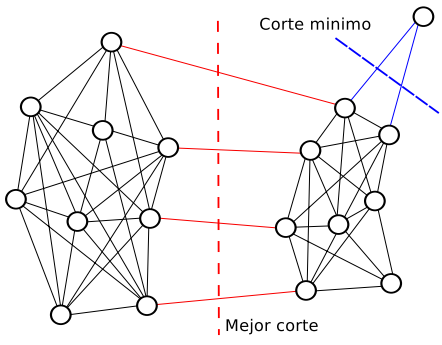


Figura 2.7: *Ejemplo de una partición errónea (Adaptada de [29])*

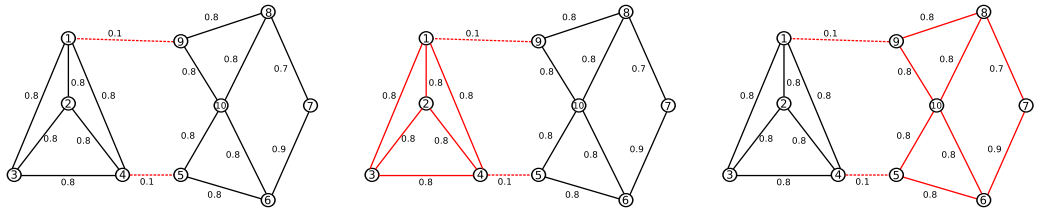
Para dar solución a este problema Shi y Malik proponen en [36] una modificación para el valor de corte $cut(A, B)$ llamado corte normalizado $Ncut(A, B)$ y demostraron que la solución óptima para este corte se encuentra en los vectores propios de la matriz Laplaciana normalizada. En particular, por el segundo menor valor propio de L_n el cual está relacionado con el punto de corte más óptimo y su respectivo vector propio con patrones similares para agruparlos en un mismo conjunto [28, 36].

$$Ncut(A, B) = \frac{cut(A, B)}{assoc(A, G)} + \frac{cut(A, B)}{assoc(B, G)} \tag{2.7}$$

Donde:

$$assoc(A, G) = \sum_{i \in A, j \in G} W(i, j)$$

$assoc(A, G)$ es la suma de los pesos entre las aristas, de A a todos los demás nodos del grafo, $assoc(B, G)$ es la suma de los pesos de las aristas de B a los demás nodos del grafo [36].



a b c

Figura 2.8: *Elementos que componen a* $Ncut(A, B)$. *a:* $cut(A, B) = 0.2$. *b:* $assoc(A, G) = 5$. *c:* $assoc(B, G) = 5$. *(Adaptado de [32])*

Demostración [36]

Dada una partición de los nodos de un grafo V en dos grupos A y B. Siendo x un vector con dimensión $N = |V|$, y con $x_i = 1$, si el nodo i está en A y $x_i = -1$, en otro caso. Con $d(i)$ como el grado del nodo i, de esta forma se puede reescribir $NCut(A, B)$ como:

$$NCut(A,B) = \frac{\sum_{x_i>0,x_j<0} -W_{ijx_ix_j}}{\sum_{x_i>0} d_i} + \frac{\sum_{x_i<0,x_j>0} -W_{ijx_ix_j}}{\sum_{x_i<0} d_i} \tag{2.8}$$

Sea D una matriz diagonal $N \times N$ con d_i en su diagonal. W una matriz simétrica $N \times N$ con $W(i,j) = w_{ij}$. Se define k como:

$$k = \frac{\sum_{x_i>0} d_i}{\sum_i d_i}$$

Sea δ un vector de unos de tamaño $N \times 1$. Usando los vectores $\frac{\delta+x}{2}$ y $\frac{\delta-x}{2}$ para $x > 0$ y $x < 0$, respectivamente, se puede reescribir $NCut(A,B)$ como:

$$NCut(A,B) = \frac{x^T(D-W)x + \delta^T(D-W)\delta}{k(\delta-k)\delta^T D\delta} + \frac{\delta(\delta-2k)\delta^T(D-W)x}{k(\delta-k)\delta^T D\delta} \tag{2.9}$$

Si se denomina

$$\alpha(x) = x^T(D-W)x; \quad \beta(x) = \delta^T(D-W)x; \quad \gamma(x) = \delta^T(D-W)\delta$$

Y

$$M = \delta^T D\delta$$

Ahora puede expresarse la ecuación 2.9 como:

$$NCut(A,B) = \frac{(\alpha(x)+\gamma(x)) + 2(\delta-2k)\beta(x)}{k(\delta-k)M} - \frac{2(\alpha(x)+\gamma(x))}{M} + \frac{2(\alpha(x))}{M} + \frac{2\gamma(x)}{M}$$

Permitiendo que $b = \frac{k}{\delta-k}$, y con $\gamma(x) = 0$, entonces:

$$NCut(A,B) = \frac{[(\delta-x) - b(\delta-x)]^T(D-W)[(\delta+x) - b(\delta-x)]}{b\delta^T D\delta}$$

Sea $y = (\delta+x) - b(\delta-x)$, se puede ver que:

$$y^T D\delta = \sum_{x_i>0} d_i - b \sum_{x_j>0} d_i$$

Ya que $b = \frac{k}{\delta - k} = \frac{\sum_{x_i>0} d_i}{\sum_{x_i<0} d_i}$ y $y^T Dy = b\delta^T D\delta$.

Uniendo lo anterior se tiene:

$$min_x Ncut(x) = min_y \frac{y^T(D-W)y}{y^T Dy}$$

Con la condición $y(i) \in \{\delta, b\} e y^T D\delta = 0$.

De forma que, si la condición sobre y se relaja y toma un valor real, se puede simplificar la expresión resultando al siguiente sistema de valores propios generalizados:

$$(D-W)y = \lambda(Dy) \tag{2.10}$$

Se considera, en primer lugar, la segunda restricción sobre y, es decir, $y^T D\delta = 0$. Se puede demostrar que satisface automáticamente la solución del sistema de valores propios.

$$D^{-1/2}(D-W)D^{-1/2} = \lambda Z$$

Donde $Z = D^{1/2}y$. Se puede verificar que $z_0 = D^{1/2}I$ es un vector propio de $D^{-1/2}(D-W)D^{-1/2}$ con valor propio igual a cero. Además, $D^{-1/2}(D-W)D^{-1/2}$ es una matriz semi definida positiva simétrica ya que $(D-W)$ es la matriz Laplaciana. Por tanto, Z_o, es el más pequeño de los vectores propios del sistema y los demás vectores propios son perpendiculares entre sí. En particular, Z_1, es el segundo vector propio más pequeño que es perpendicular a Z_0. Se tiene que:

- $y_0 = 1$ es el vector propio más pequeño con valor propio igual a cero.
- $0 = z_1^T$, $z_0^T D\delta$, donde y_1 es el segundo vector propio más pequeño de la ecuación 2.10.

Dada una matriz simétrica real A, bajo la restricción de que x es ortogonal a los $j-1$ vectores propios más pequeños $[x_1, \ldots, x_{j-1}]$, el cociente $\frac{x^T Ax}{x^T x}$ se minimiza con el vector propio próximo más pequeño a x_j y su valor mínimo es el valor propio λj.

Aplicando los anteriores criterios:

$$Z_1 = arg \cdot min_{Z^T Z_0 = 0} \frac{Z^T D^{-1/2}(D - W)D^{-1/2}}{Z^T Dy}$$

Por tanto:

$$y_1 = arg \cdot min_{y^T D\beta = 0} \frac{Y^T (D - W)y}{Y^T Dy}$$

De manera que el segundo vector propio más pequeño del problema de vectores propios generalizados es la solución real a la relajación del problema *normalized cut*.

En resumen, se ha demostrado como el criterio de *normalized cut*, puede resolver el problema de agrupamiento mediante la división de grafos. Además, se muestra cómo el criterio impuesto puede calcularse eficientemente resolviendo un sistema de valores propios [36].

Finalmente se presenta el pseudocódigo de un algoritmo clásico de agrupamiento espectral que en esencia está compuesto por los temas abordados hasta el momento.

Algoritmo 1: Agrupamiento Espectral

Input: Imagen I, Número de grupos k.
Output: Imagen segmentada.

// Construir el vector de características.
for all píxel de la imagen I $[var : p_{xy}]$ **do**
    $P(p_{xy}) = [\ L \quad a \quad b \quad x \quad y\ ]$ Los valores de color y posición del píxel $I(x, y)$.
end for
// Calcular la matriz de adyacencia totalmente conectada.
for all fila de P $[var : i]$ **do**
    for all fila de P $[var : j = j + i]$ **do**
        $A_f(i, j) = A_f(j, i) = exp^{-\left\|x_i - x_j\right\|^2 / 2\sigma^2}$ Calcular la distancia entre los píxeles.
    end for
end for
// Calcular la matriz Diagonal.
for all fila de la matriz de Afinidad A $[var : i]$ **do**
    $sum = sum(A.row(j))$ Sumar los valores de sus columnas.

14: $D(j, j) = sum$ Ubicar en la posición de su diagonal.
15: **end for**
16: // Calcular la matriz Laplaciana.
17: $L_n = D^{-1/2} \cdot A \cdot D^{-1/2}$
18: // Calcular los vectores y valores propios.
19: Obtener valores propios con función *eigen*
20: Ordenar de forma ascendente los vectores propios; según el valor propio.
21: Determinar los k vectores propios más pequeños; no tener en cuenta el primer vector propio.
22: // Etiquetado y Reconstrucción de la imagen.
23: Etiquetado de los píxeles con K-Means.
24: **for all** fila de la imagen $I\ [var : i]$ **do**
25: **for all** fila de la imagen $I\ [var : j]$ **do**
26: $I(i, j) = [\ L_k \quad a_k \quad b_k\ ]$ Asignar un color según el grupo para el píxel $I(i, j)$.
27: **end for**
28: **end for**

La explicación del algoritmo K-Means es realizada en la sección 2.4.1; ya que ahí tiene mayor relevancia.

En la Figura 2.9 se presenta gráficamente los pasos que conforman el desarrollo del algoritmo aplicando el criterio de corte normalizado. Se observa la construcción de la matriz de similaridad con base en la función de núcleo gaussiana, la cuál presenta inconvenientes por la selección del criterio de ponderación de decrecimiento de las afinidades σ estático, por tanto la medida disminuye su redimiendo cuando se expone al algoritmo frente a diferentes tipos de datos, por este motivo en la Sección 2.3 se presenta un mejor criterio para otorgar las afinidades, además, las construcción de la matriz presenta un crecimiento exponencial de sus dimensiones cuando aumentan la cantidad de datos a ser evaluados; en la Sección 2.4 se presenta un método para reducir la cantidad de datos que conforman la matriz.

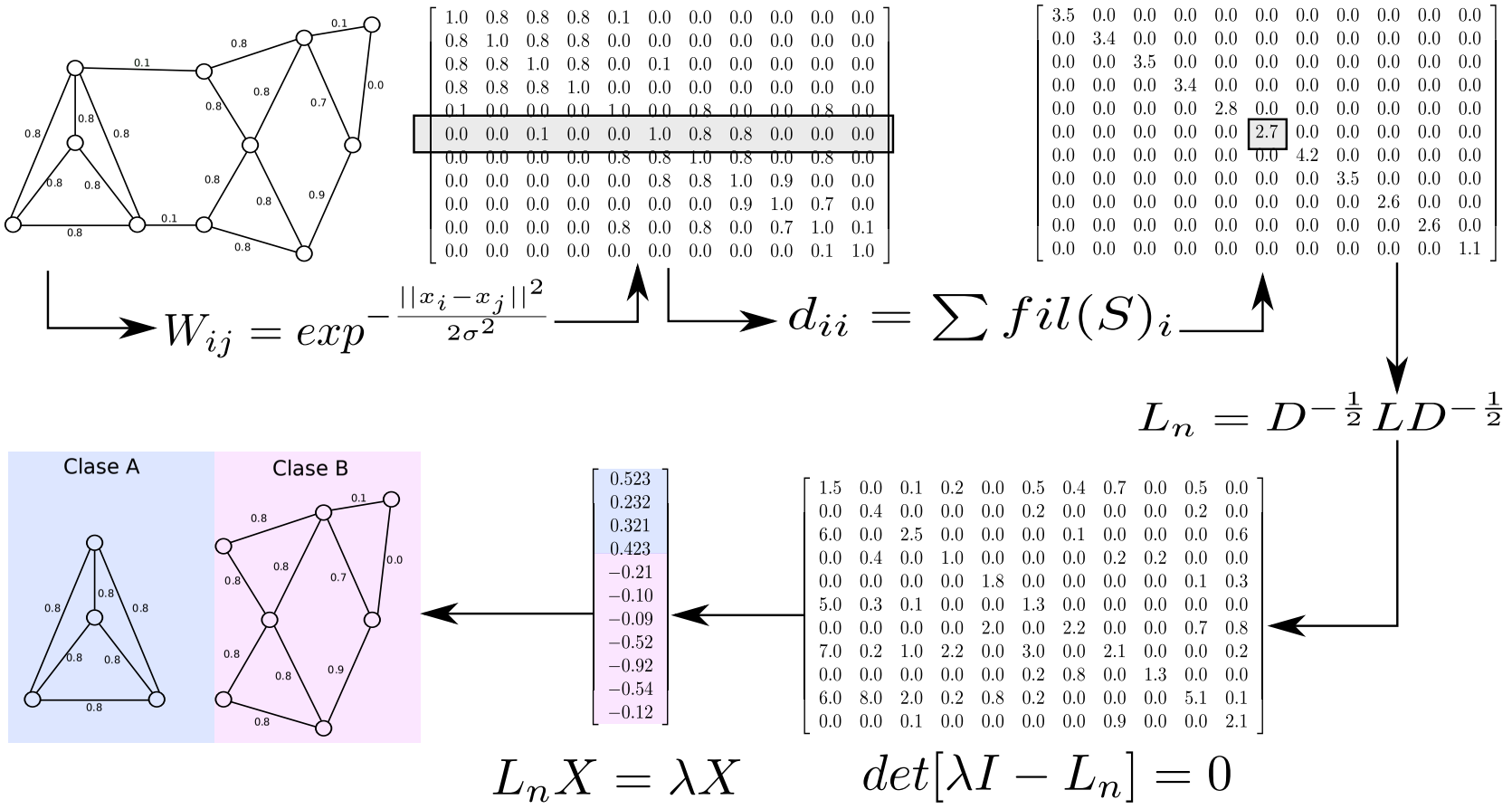


Figura 2.9: *Resumen corte normalizado*

2.3. Agrupamiento Difuso

En el agrupamiento clásico cada píxel debe ser asignado a un solo grupo. El análisis con agrupamiento difuso relaja este requerimiento permitiendo pertenencias graduales y ofreciendo la oportunidad de trabajar con píxeles que pertenecen a más de un grupo al mismo tiempo.

Un conjunto difuso puede definirse como:

$$C_f = \{(x, \mu_C(x))|x \in X\} \tag{2.11}$$

Donde

- $\mu_c(x) \rightarrow [0, 1]$ es la función de pertenencia.
- $X = \{x_1, x_2, \ldots, x_N\}$ Es el universo de discurso, en el cual $x_k \in \Re^p$ con $k = 1, 2, \ldots, N$ y p denota el número de atributos que definen a cada elemento de X, con frecuencia el elemento x_k es llamado vector de características.
- C es un número natural que define la cantidad de grupos en que será dividido X. Restringido por: $2 \leq C < k$. Esto para garantizar una partición lógica [43].

Un píxel puede ser representado por un vector de características x_k donde los p atributos pueden ser los componentes RGB como también su posición espacial (x, y) de esta manera al hablar de un vector de características será equivalente a hablar de un píxel.

2.3.1. Matriz de partición

Sea U una matriz con dimensión $C \times N$, en la cual las filas representan la cantidad de grupos difusos, y las columnas los elementos del conjunto X.

$$U = [u_{ij}]_{C \times N} = \begin{bmatrix} u_{11} & u_{12} & \ldots & u_{1N} \\ u_{21} & u_{22} & \cdots & u_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ u_{C1} & u_{C2} & \ldots & u_{CN} \end{bmatrix}$$

u_{ij} denota el grado de pertenencia del elemento x_k al i-ésimo grupo C, ya que $\mu_{c_i}(x)$ puede tomar cualquier valor real entre 0 y 1, se deben cumplir las siguientes dos condiciones para garantizar una partición difusa.

$$\sum_{i=1}^{C} u_{ij} = 1,\ \forall j \in [1, N]$$

Lo cual quiere decir que un vector característico x_{jk} tiene su valor de membresía 1 dividido entre todos los grupos.

$$1 < \sum_{j=1}^{N} u_{ij} < N,\ \forall i \in [1, C]$$

Indica que la suma de grados de pertenencia de vectores de características en un grupo dado no excede el número total de vectores de características.

Considerando a U^t como el valor de la matriz de partición y V^t como el valor del vector de prototipos en la iteración t-ésima del método, se presenta a continuación los pasos para construir la matriz U

1. Definir el número de particiones C.

2. Cargar la matriz U^0 con valores aleatorios $[0 - 1]$.

3. Definir el vector $V = \{v_1, v_2, \ldots, v_c\}$. Los elemento de V serán los centroides o patrones de cada grupo, estos centroides también se escogen de manera aleatoria por lo que en primera instancia son llamados prototipos.

4. Optimización iterativa de la siguiente función objetivo:

$$J_m(U, V) = \sum_{i}^{C} \sum_{j}^{N} u_{ij}^m \left\| x_j - V_i \right\|^2$$

   Dónde $\|*\|$ es la distancia euclidiana que expresa la similitud entre cualquier punto de datos medidos y el prototipo. m es el exponente de ponderación, $1 < m < \infty$, con $m = 1$, se tiene una partición clásica, a medida que m aumenta se disminuye la rigidez en las particiones.

5. Se intenta minimizar la distancia entre los datos y el centro del grupo, esto se logra recalculando dinámicamente los pesos u_{ij} y los centroides prototipo utilizando las ecuaciones 2.12 y 2.13 hasta cuando el criterio J_m alcance un valor estacionario que representa un mínimo local o punto de silla.

$$u_{ij} = \left[\sum_{p=1}^{C} \left(\frac{\|x_j - v_i\|}{\|x_k - x_p\|} \right)^{1/(\mathrm{m}-1)} \right]^{-1}, \; 1 \leq i \leq C, \; 1 \leq j \leq N \tag{2.12}$$

$$V_k = \frac{\sum_{j=1}^{N} u_{ij}^m x_j}{\sum_{j=1}^{N} u_{ij}} \; 1 \leq i \leq C \tag{2.13}$$

   Como se puede observar estas ecuaciones dependen una de otra. El método más usado para aproximar una solución a estas ecuaciones es el de iteración de Picard [50], que consiste en resolver las ecuaciones en forma iterativa. En otras palabras, se fija v y se optimiza U, luego se fija U y se optimiza v.

6. Verificar que $|U^{t+1} - U^t| < \varepsilon$ de lo contrario volver al paso 4 [15, 43]. En cada interacción los centroides prototipo de cada grupo son reubicados al igual que el grado de pertenencia de los datos a estos. Se considera que los centroides proporcionan una buena distribución de los datos cuando la variación entre U^{t+1} y U^t es mínima, es decir menor que ε. En algunos casos la condición para detener la optimización de j_m es fijar el número máximo de iteraciones.

2.3.2. Matriz de similaridad

La matriz de partición U contiene toda la información requerida sobre la proximidad relativa de cada píxel caracterizado por x_k con todos los prototipos V_i, $i = 1, 2, \ldots C$. El valor de pertenencia en la matriz de partición denota la probabilidad de que un grupo de píxeles pertenezca a un grupo específico. Por lo tanto, a través de comparar el grado de pertenencia de dos puntos de datos a un grupo, se puede suponer razonablemente si estos son similares o no, ya que; dos píxeles muy similares tendrán grado de pertenencia cercano a uno, por el contrario valores cercanos a cero, reflejan poca similitud entre estos. Por lo tanto la construcción de la matriz de similaridad difusa se realiza a partir de las siguientes reglas [15]:

$$S_{ij} = \begin{cases} 1 & Si\,(x_i, x_j)\,pertenecen\,a\,la\,misma\,vecindad \\ max[u_{li}, u_{lj}],\ 1 \leq l \leq t & Si\,(x_i, x_j)\,pertenecen\,a\,la\,t\,vecindad\,próxima \\ 0 & en\,cualquier\,otro\,caso \end{cases}$$

2.4. Súper Píxeles

La segmentación de imágenes basada en redes complejas presenta una limitación. Una imagen de tamaño $M \times M$ es mapeada a una red compuesta por M^2 nodos, donde cada píxel de la imagen es mapeado como una red, y el peso de la relación se establece proporcional a la similaridad entre píxeles [37]. En la presente sección se pretende explicar cómo utilizar súper píxeles y como éstos proporcionan una eficiente y compacta representación de la red.

En el procesamiento de imágenes es usual realizar una etapa de pre-procesamiento para disminuir la cantidad de datos a procesar, en consecuencia se presenta una disminución en el consumo de recursos computacionales en tareas posteriores. Súper píxel es un enfoque de segmentación de imágenes basado en regiones; que consiste en particionar una imagen en varios grupos de píxeles [38]. Esta técnica puede ser usada como una pre-segmentación para reducir una cantidad de píxeles de la imagen o como una segmentación final. Para que una técnica sea útil, esta deber ser rápida, fácil de usar y producir segmentaciones de buena calidad. Usualmente los súper píxeles son convexos y de tamaño casi uniforme, como se muestra en la Figura 2.10

a b

Figura 2.10: *Convexidad y tamaño uniforme para la segmentación con súper píxeles. a: y b: representan la segmentación de una imagen con diferente cantidad de súper píxeles. (Tomada de [24])*

Para la segmentación de imágenes por medio de súper píxeles se han desarrollado varias

técnicas. Entre los algoritmos más destacados se tienen: *Simple Linear Iterative Clustering* (SLIC) y *Speed-Up Turbo Píxel* (STP). SLIC como STP se basan en el algoritmo K-Means como parte primordial. En resumen, estas dos técnicas hacen una partición inicial, segmentar en rectángulos. Seguidamente se realiza el intercambio o asignación de píxeles a los centroides que representan cada rectángulo.

En las subsecciones siguiente se trabajarán los conceptos y algunos algoritmos de súper píxel. Primero se explica K-Means como pilar principal del algoritmo SLIC y STP. Posteriormente se tiene CIELab, como el sistema de colores que se trabaja sobre el algoritmo SLIC. Con estas ideas claras se procede a explicar el funcionamiento tanto de SLIC, como de STP; que son las subsecciones finales.

2.4.1. K-Means

En la Figura 2.11 se puede visualizar los pasos del agrupamiento K-Means.

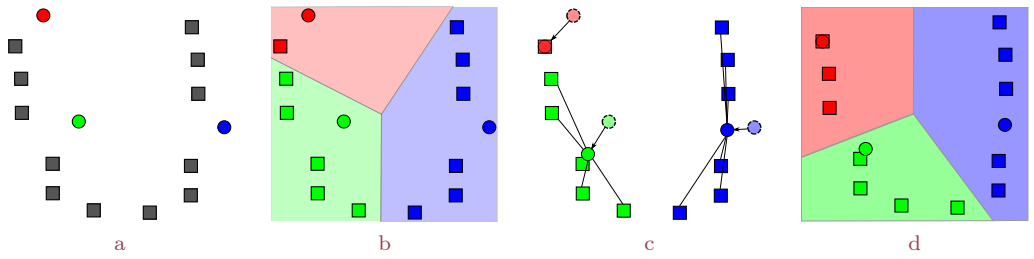


Figura 2.11: *Pasos de segmentación del algoritmo K-Means. a: $k = 3$ centroides r_k iniciales seleccionados aleatoriamente. b: Asignar los puntos a cada centroide r_k. c: Repetir a: y b: hasta obtener los grupos finales. (Tomada de https://en.wikipedia.org/wiki/K-means_clustering)*

Sea $X = \{x_1, \ldots, x_n\}$ un conjunto de n puntos $d_{dimensionales}$ para ser agrupados en k grupos, $C = \{C_i | i = 1, \ldots, k\}$. El algoritmo busca una partición tal que minimice una función de ajuste; las medidas pueden ser: la distancia Euclídea, similitud coseno, etc [48, 49]. Para este caso se tomará como medida, el error cuadrático. La medida será dada entre el centroide del grupo y los respectivos puntos asociados. La función de optimización es descrita en la ecuación 2.14.

$$J(C) = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - r_i\|^2 \tag{2.14}$$

donde:

$$r_i = \frac{1}{n_{c_i}} \sum_{x \in C_i} x_i$$

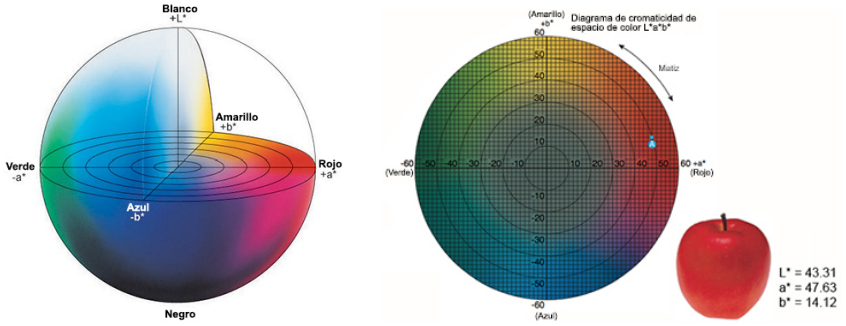
r_i es el centro del grupo C_i. n_{C_i} representa la cantidad de píxeles presentes en el grupo C_i. Inicialmente el algoritmo selecciona aleatoriamente k centros, seguidamente se asignan los píxeles x_i al centro más cercano a modo de reducir el error cuadrático, finalmente se recalculan los centros r_i. Las dos anteriores operaciones se repiten hasta obtener una variación mínima en los centros o hasta alcanzar el máximo número de iteraciones [39].

Las desventajas que presenta el algoritmo K-Means, son la sensibilidad al ruido y los *outliers*[1], además no puede ser usado para encontrar grupos con forma arbitrarias.

2.4.2. CIELab

CIELab es el segundo de los dos sistemas adoptados por la CIE en 1976 como modelos que mostraron una separación uniforme de los colores en el valor; porque correlaciona los valores numéricos de color consistentemente con la percepción visual humana. CIELab es un sistema de color oponente basado en el sistema de Richard Hunter anteriormente llamado L, a, b. La oposición de color se correlaciona con los descubrimientos en la década de 1960; que en algún lugar entre el nervio óptico y el cerebro, los estímulos en la retina por el color se convierten en distinciones entre claro y oscuro, rojo y verde; y azul y amarillo [40]. CIELab indica estos valores con tres ejes: L*, a*, y b* como se visualiza en la Figura 2.12.

[1]Es un punto que cae muy lejos del centro del grupo más cercano.



a b

Figura 2.12: *Modelo de color CIELab. a: Oposición de color: Claro-Oscuro, Rojo-Verde y Azul-Amarillo. b: Medición de los valores* [L a b] *para una manzana. (Tomada de* htttp://www.sensing.konicaminolta.com.mx/2014/09/entendiendo-el-espacio-de-color-cie-lab/*)*

De la Figura 2.12 se tiene que: $L = luminosidad(0-100)$, $a = coordenadas\, rojo/verde$ y $b = coordenadas\, amarillo/azul$.

2.4.3. Simple Linear Iterative Clustering (SLIC)

Achanta en [41] introduce un algoritmo denominado SLIC para la extracción de súper píxeles en imágenes de color, considerando el modelo de colores CIELab. El enfoque del algoritmo es desarrollado con base al algoritmo K-Means, además emplea un espacio de búsqueda más eficiente; que permite reducir el costo computacional.

El algoritmo consta de los siguientes pasos:

1. Inicialmente, una imagen de N píxeles es dividida en k_{sp} regiones rectangulares, o sea, súper píxeles de dimensión $S \times S$, donde $S = \sqrt{\frac{N}{k}}$. Cada súper píxel i es representado por un centroide definido como un vector de cinco dimensiones $Ci = [\ L_i \quad a_i \quad b_i \quad x_i \quad y_i\ ]^T$, donde L , a y b son los valores medios de los tres componentes del modelo CIELab. Además, x y y son las coordenadas del súper píxel i.

2. Los k centroides son trasladados a una nueva posición, la que presente el menor valor de gradiente. El valor de gradiente se calcula en un vecindario de 3×3 con respecto al centroide primeramente definido. Esto se hace para evitar centrar un súper píxel en un borde, y para reducir la posibilidad de sembrar un centroide en un píxel ruidoso.

Los gradientes de la imagen son computados como muestra la siguiente ecuación:

$$G(x,y) = |I(x+1,y) - I(x-1,y)|^2 + |I(x,y+1) - I(x,y-1)|^2$$

Donde $I(x,y)$ es el vector L, a y b correspondiente a un píxel en la posición (x,y). $|.|$ es la norma L_2 o norma Euclidiana.

3. En cada iteración el píxel j es asociado con el centroide del súper píxel más cercano; dado que el área de búsqueda de cada súper píxel incluye a j. El tamaño del área de búsqueda es dado por $2S \times 2S$ como muestra la Figura 2.13. En seguida, se calculan los nuevos centroides C_j, considerando la media de todos los píxeles del súper píxel i. Este proceso se ejecuta a través de la minimización de la función D, definida por la ecuación 2.15:

$$D = \sqrt{d_c^2 + \left(\frac{d_s}{S}\right)^2 m^2} \tag{2.15}$$

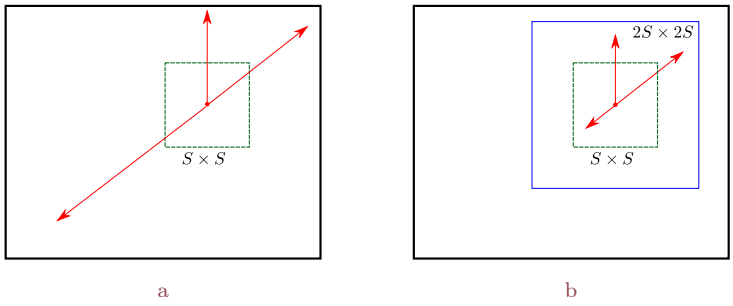


Figura 2.13: *Reducción en la región de búsqueda. a: K-Means (tradicional) busca en la imagen completa. b: K-Means para SLIC busca en una región acotada. (Adaptada de [24])*

Donde d_c es la distancia en el espacio de colores y d_s es la distancia espacial, representadas, respectivamente por las siguientes ecuaciones:

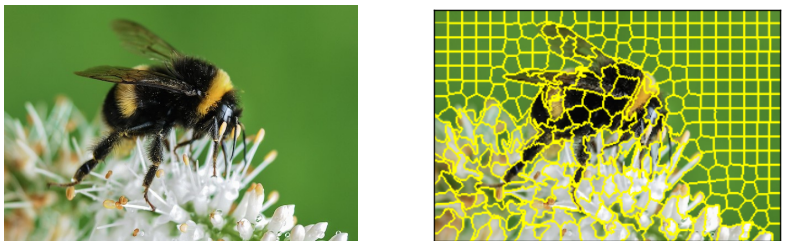
$$d_c = \sqrt{(L_i - L_j)^2 + (a_i - a_j)^2 + (b_i - b_j)^2}$$

$$d_s = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

Se sugiere que los súper píxeles exactos pueden ser obtenidos con un máximo de 10 iteraciones. El parámetro $m(1 \leq m \leq 20)$ controla la compacidad de los súper

píxeles, o sea, cuanto mayor sea el valor de m mayor es el énfasis en la proximidad espacial, obteniendo grupos más compactos.

4. Finalmente, los grupos de píxeles aislados pueden surgir después de terminar las interacciones. Estos grupos son asociado al mejor súper píxel vecino. En la Figura 2.14 se visualiza el resultado de un agrupamiento por SLIC.



a b

Figura 2.14: *Segmentación de imagen con SLIC. a: Imagen original. b: Imagen segmentada; k=400. (Realizada por los autores)*

2.4.4. Speed-up Turbo Píxel

En [42] se propone una técnica para la generación de súper píxeles basado en el algoritmo de agrupamiento K-Means, capaz de generar súper píxeles uniformes con bajo costo computacional, llamado STP. Inicialmente, se divide la imagen en regiones rectangulares de acuerdo al número de súper píxeles deseados, con centros equidistantes entre la imagen. En el siguiente paso, los píxeles de los límites de las regiones son asignados a nuevas regiones minimizando la función de costo indicada en ecuación 2.16:

$$C_{x,y}(i) = |I(x,y) - I_i| \{T_1\} + \lambda_2 \left|(x - C_x^i)^2 + (y - C_y^i)^2\right| \{T_2\} \tag{2.16}$$

Donde I_i indica la intensidad media de el i^{th} segmento, x y y son la ubicación del píxel de prueba entre diferentes segmentos. C_x^i y C_y^i son las coordenadas del centroide del i^{th} segmento. λ_1 corresponde a la ponderación de la similitud de intensidad y las restricciones de conectividad. T_1 garantiza la similaridad de color de dos píxeles que serán agrupados. T_2 garantiza la convexidad de dos píxeles, evitando que dos píxeles muy distantes sean agrupados. La medida de la convexidad puede ser modificada cambiando el parámetro $\lambda 2$.

En el proceso de actualización de los súper píxeles, se prueban los píxeles individuales

situados en los límites del segmento, lo que asegura la conectividad de los súper píxeles. El algoritmo es rápido, ya que solo requiere probar un número limitado de píxeles entre segmentos vecinos (número máximo de vecinos 4). Una vez que se han probado todos los píxeles del contorno, la intensidad media del súper píxel y la ubicación del centroide, se actualizan. Cuando el número de píxeles intercambiados está por debajo de un límite, las iteraciones se detienen y se finaliza la generación del súper píxel. En la Figura 2.15, se visualiza los súper píxeles obtenidos por STP.

a b

Figura 2.15: *Segmentación de imagen con STP. a: Imagen original. b: Imagen segmentada; k=400. (Realizada por los autores)*

Después de realizar la implementación de los algoritmos SLIC y STP, y de comparar los resultados obtenidos por estos (ver Figura 2.14 y 2.15) se concluye que el algoritmo SLIC genera súper píxeles más compactos, además su implementación es más simple, lo que conlleva a una menor complejidad algorítmica. Por estos motivos se va a hacer uso del algoritmo SLIC para dar solución a la propuesta que se desarrolla en este trabajo.

Capítulo 3

Diseño del algoritmo.

En este capítulo se presenta en detalle el desarrollo del algoritmo SFSC (*Super pixel, Fuzzy and Spectral Clustering*), donde se puede diferenciar y analizar detalladamente los pasos que han sido implementados en el algoritmo. Además, se puede observar el desarrollo de una interfaz que permite la visualización y evaluación de los algoritmos a comparar.

Para llevar a cabo la implementación de la propuesta se ha usado el lenguaje de programación C++, se aprovecha la potencia de librerías para trabajo matricial como lo son *eigen* y *armadillo*; para optimización como son intel® MKL (*Math Kernel Library*), *Lapack*, *Blas* y *OpenMp*; además, se trabajó con *OpenCV* para manejo de las imágenes. Para el desarrollo de la interfaz gráfica se aprovechó el gran potencial del framework multi plataforma *QT*. Tener en cuenta que la instalación y configuración de las librerías se encuentra detallado en el manual de usuario del algoritmo presente en los anexos.

3.1. Algoritmo de Segmentación

El algoritmo consta de la siguientes etapas: obtener la matriz de características de la imagen a segmentar, pre-segmentación o diezmado de la imagen con el algoritmo *Simple Linear Iterative Clustering* (SLIC), conformar la nueva matriz de características; procedente de los centroides de los grupos o súper píxeles de SLIC, generar la matriz de similaridad con base a una función difusa, obtener la matriz Laplaciana, determinar el o los k vectores propios correspondientes al o los k valores propios más pequeños y finalmente se implementa un algoritmo de clasificación, *K-Means*; para determinar a qué grupo corresponde cada súper

píxel.

En la Figura 3.1 se aprecia claramente las modificaciones que se realizaron al algoritmo clásico de agrupamiento espectral, asimismo se describen y se resaltan los pasos que conforman estos algoritmos. En las siguientes subsecciones se procede a explicar detalladamente cada uno de los bloques del diagrama para el algoritmo SFSC.

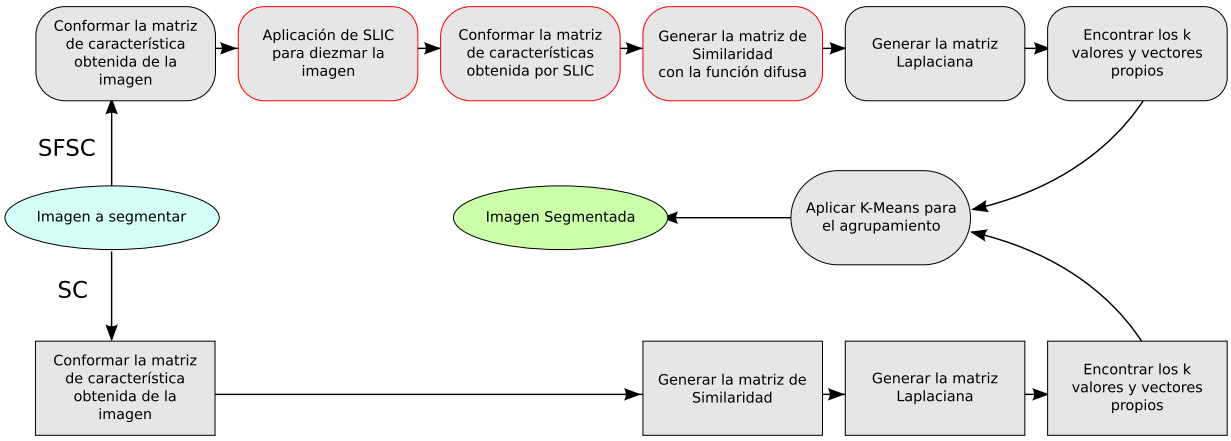


Figura 3.1: *Comparación gráfica entre el algoritmo de agrupamiento clásico SC y el algoritmo SFSC. (Realizada por los autores)*

3.1.1. Construcción matriz de características

El píxel puede definirse como la más pequeña de las unidades homogéneas de color que componen una imagen de tipo digital, un píxel puede ser caracterizado por medio de un vector según el modelo de color escogido (Grey, CIElab, RGB, CMYK, HSV, etc). En el presente trabajo se ha seleccionado el modelo de colores CIElab debido a su aceptación en la comunidad científica; puesto que sus propiedades no se ven afectadas por el dispositivo donde se reproducen. Para la construcción de la matriz de características se tiene en cuenta los componentes de color (L, a, b) y se adicionan dos características que proporcionan la ubicación espacial (x, y) del píxel. La Figura 3.2 ilustra la posible representación del vector característico de un píxel ubicado en la posición (i, j). Además se va a utilizar la imagen de la Figura 3.2 de tamaño (481×321) que simbólicamente será escrita como $(m \times n)$, para representar cada paso de la segmentación. Se segmenta en dos grupos, se pre-segmenta en 21 súper píxeles y se crea una matriz difusa de dos grupos.

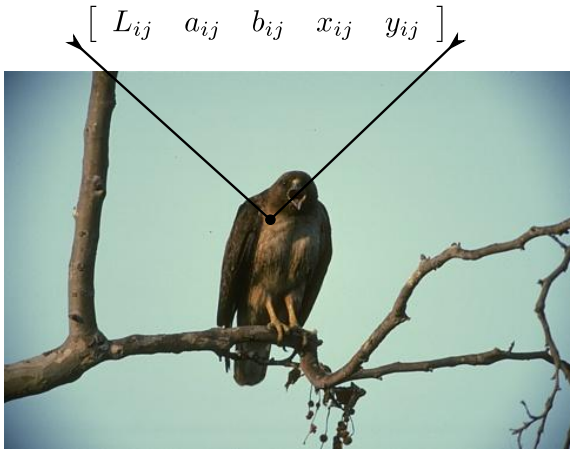


Figura 3.2: *Caracterización de una imagen. L=Luminosidad* $(0 \rightarrow 100)$, *a= Coordenadas Rojo/Verde, b = Coordenadas Amarillo/Azul y* (x, y) *= Ubicación e espacial. (Realizada por los autores)*

En el desarrollo del algoritmo se ha utilizado la librería *OpenCV* para obtener la matriz característica de la imagen basada en el criterio de color seleccionado. En primera instancia se obtiene una matriz de características en *BGR* (Blue, Green, Red) con tipo de dato Uint8; que es la representación que por defecto utiliza *OpenCV* cuando carga la imagen. Se hace uso de la función cvtColor y como parámetro de conversión *BGR2lab* para llevar al espacio de colores deseado. Se debe tener en cuenta que el algoritmo trabaja con valores punto flotante, por este motivo se hace la conversión a un tipo de dato Float32, con 4 bytes de precisión. En la Figura 3.3 se representa las capas de la imagen en *Lab*.

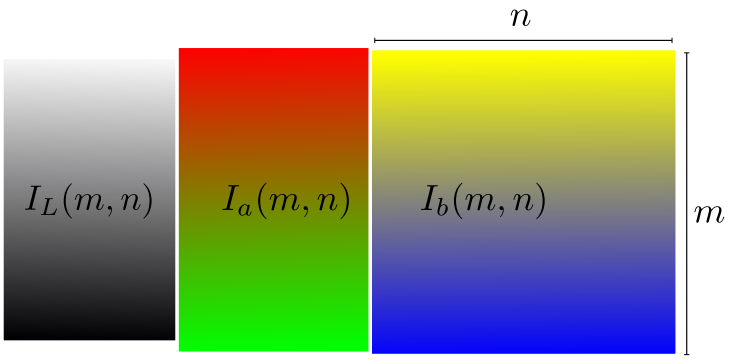


Figura 3.3: *Representación de las capas de una imagen en espacio de colores Lab. (Realizada por los autores)*

En este punto no es necesario hacerle ninguna modificación estructural a la matriz que contiene la imagen, debido a que se puede extraer la información necesaria de la imagen

obtenida y convertida al espacio de colores *Lab*. Ahora se presenta el pseudocódigo de la construcción de la matriz de características descrita anteriormente.

Algoritmo 2: Matriz de características

Input: Imagen Original (I_o).

Output: Matriz características (Imagen Lab) (I_{lab}).

1: Cargas imagen.

2: Conversión I_o a espacio de colores *Lab*.

3: Conversión I_{lab} a $float32$.

3.1.2. Diezmado de la imagen

En este paso se construye la matriz $C = [c_0 c_1 \dots c_{k_{sp}}]^T$, cuyas filas son el número de centros equivalente a los k_{sp} súper píxeles en que se va a representar la imagen, y las columnas son las características del píxel centroide c_i. La ecuación 3.1 define la conformación de la matriz C para la imagen muestra, con 21 súper píxeles.

$$C = \begin{bmatrix} L_0 & a_0 & b_0 & x_0 & y_0 \\ L_1 & a_1 & b_1 & x_1 & y_1 \\ \vdots & & \cdots & & \vdots \\ L_{20} & a_{20} & b_{20} & x_{20} & y_{20} \end{bmatrix} \tag{3.1}$$

Seleccionados los centroides, se construyen dos matrices de la misma dimensión que la imagen de tamaño$(m \times n)$, que contendrán las etiquetas L_a y las distancias D. Estas matrices serán actualizadas durante el proceso de segmentación, la matriz D almacena la menor distancia entre el píxel a analizar (i, j) y el respectivo centroide c_i, al determinar la menor distancia se actualizada L_a con el identificador del súper píxel c_i en la posición (i, j). Con L_a se puede recalcular los nuevos centroides, además permite observar la segmentación final en súper píxeles. Como paso medio, se tiene el cumplimiento de conectividad en el cual se eliminan regiones aisladas y/o súper píxeles que contienen baja cantidad de píxeles; con esto se busca que las regiones queden más compactas, por este motivo es necesario recalcular el número de súper píxeles k_{sp}, esto implica el determinar una nueva matriz C. En la Figura 3.4 se representan los pasos que ejecuta el algoritmo SLIC.

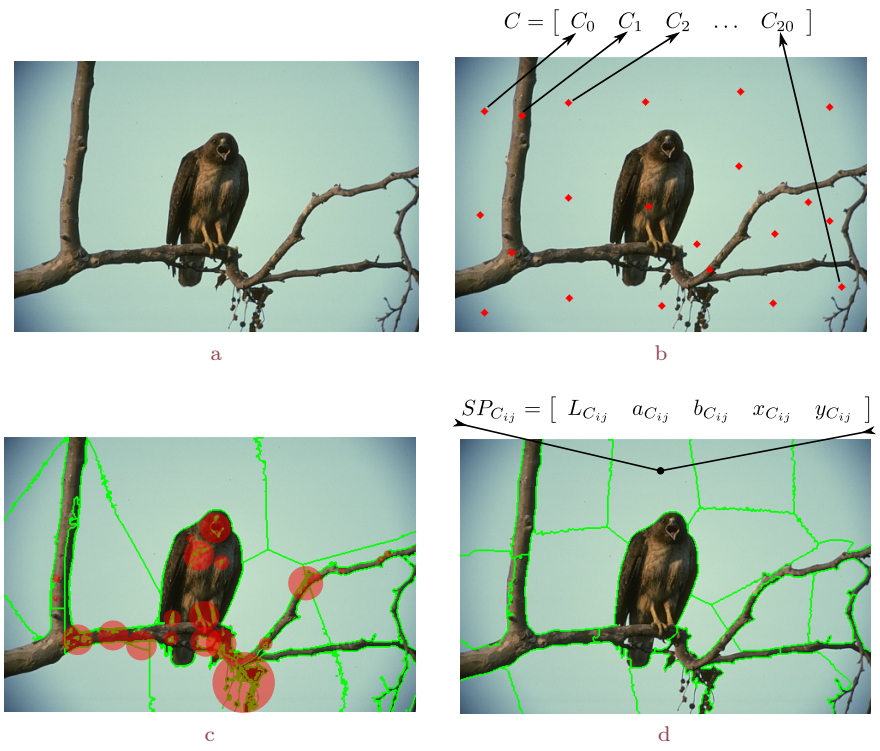


Figura 3.4: *Pasos para segmentar una imagen con SLIC. a: Imagen Original. b: Ubicación de los k centroides. c: Penalización de regiones aisladas. d: Imagen segmentada. (Realizada por los autores)*

La imagen original es representada por $154.401e^3$ píxeles, mientras que la imagen tratada con SLIC está representada por 21 súper píxeles como se muestra en la Figura 3.5, ahora se puede observar una reducción significativa de datos a procesar por agrupamiento espectral. La matriz C contiene el valor medio de las componentes Lab y posición de los píxeles pertenecientes a cada súper píxel k_{sp}.

Figura 3.5: *Imagen representada por 21 súper píxeles. (Realizada por los autores)*

De aquí en adelante el trabajo de segmentación por agrupamiento espectral se realiza con la matriz de características de súper píxel C. A continuación se presenta el pseudocódigo del algoritmo SLIC.

Algoritmo 3: SLIC

Input: Matriz características I_{lab}, Número súper píxeles (k_{sp}), Factor de ponderación (m_{sp}).

Output: Vector característico ($C_{[k_{sp}\times 5]}$).

// Calcular el tamaño del paso de rejilla, S.

$N = m * n$;

$S_t = \sqrt{\frac{N}{k_{sp}\cdot\frac{\sqrt{3}}{2}}}$; $n_c = round(\frac{columnas}{S_t - 0.5})$;

$S = round(\frac{columnas}{n_c + 0.5})$

$n_f = round(\sqrt{\frac{filas}{\frac{\sqrt{3}}{2\cdot S}}})$; $v_s = \frac{filas}{n_f}$

// Inicializar los centros

$c_i = [L, a, b, x, y]^T$, por cada píxel muestra agregar un paso de rejilla regular S.

// Perturbar los centros de los grupos en un vecindario de tamaño $m \times m$. El centro se obtiene del vecino que presente el menor valor de gradiente.

$r = v_s/2$

for all c_i en la posición x $[var : p_x]$ **do**

  if px_x es par **then**

    $p = \frac{S}{2}$

  else

    $p = S$

  end if

  for all c_i en la posición y $[var : p_y]$ **do**

    $p = round(p)$; $r = round(r)$

    for $(p-1) \rightarrow (p+2)$ y $(r-1) \rightarrow (r+2)$ **do**

      $G(p_x, p_y) = ||I(p_x + 1, p_y) - I(p_x - 1, p_y)||^2 + ||I(p_x, p_y + 1) - I(px_x, px_y - 1)||^2$

      El píxel que contenga el menor valor de gradiente sera el nuevo centroide c_j

      $p+ = S$

    end for

    $r+ = v_s$

  end for

end for
// Calcular los súper píxeles
for $1 \rightarrow 10$ **do**
  for all Grupo con centro c_i **do**
    // Asignar los mejores píxeles coincidentes desde una ventana $2Sx2S$ alrededor de centro, de acuerdo a la medida de la distancia. d_{lab} (Distancia entre componentes de color). d_{xy} (Distancia pondera entre la posición de los píxeles)
    for all píxel(p_x) $(c_i(x) - S > p_x > c_i(x) + S)$ y $(c_i(y) - S > p_x > c_i(y) + S)$ **do**
      $d = d_{lab} + (m/S) \cdot d_{xy}$; Calcular la distancia entrepx y c_i
      if $d <$ *anterior distancia* **then**
        $L_a(p_x(i,j)) = K_c$
        $D(p_X(i,j)) = d$
      end if
    end for
  end for
  // Calcular los nuevos centroides.
  $c_i = {}^{T_{c_i}}/_{T_{px}}$; Cantidad de píxeles T_{px} que quedaron asignados al centroide c_i. Sumatorio de los componentes de cada píxel perteneciente a c_i; $T_{c_i} = [T_L, T_a, T_b, T_x, T_y]$.

end for
Cumplir conectividad

La conformación de la matriz característica presentada como paso 3 de SFSC en la Figura 3.1, se da solución en el algoritmo SLIC al constituir la matriz C.

3.1.3. Creación de la matriz prototipo

Para realizar las segmentación por agrupamiento espectral es necesario construir la matriz de similaridad S, para su desarrollo se aplicará como medida una distancia difusa. El primer paso es la construcción de la matriz prototipo difusa U que contendrá las relaciones parciales entre los súper píxeles C y los grupos a segmentar finalmente k_f. La matriz V contiene los centroides de los grupos. Cabe destacar que el desarrollo del algoritmo es basado en *Fuzzy C-Means* (FCM). En la Figura 3.6 se observa las relaciones difusas que existe entre los súper píxeles y los grupos a segmentar k_f; en este caso dos grupos.

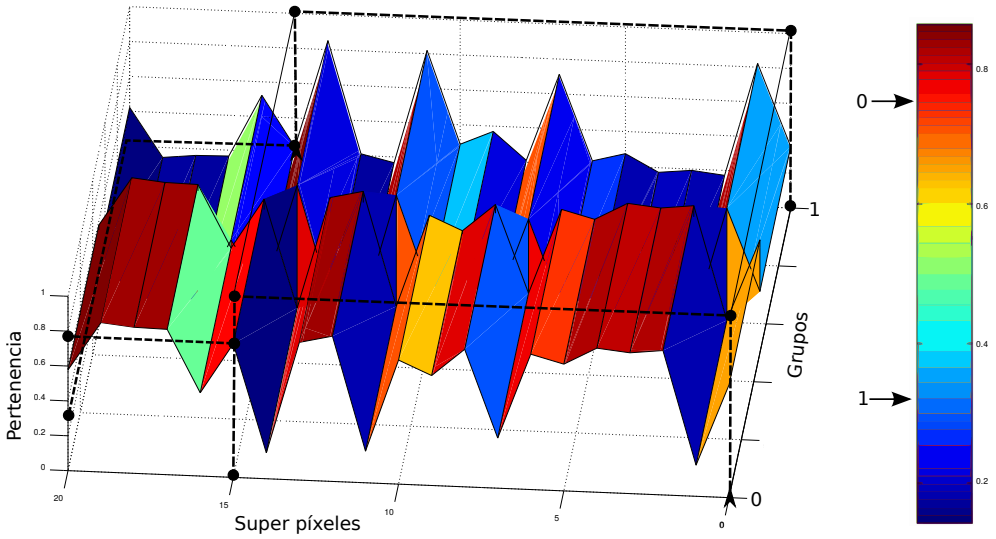


Figura 3.6: *Grados de pertenencia de los súper píxeles a los grupos. Ejm: Para el súper píxel 15 se tiene un grado de pertenencia para el grupo* $0 \simeq 0.78$ *y para el grupo* $1 \simeq 0.35$. *(Realizada por los autores)*

A continuación se presenta el pseudocódigo del algoritmo.

Algoritmo 4: Matriz prototipo de agrupamiento difuso

Input: Vector característico (C), Número de grupos (k_f), Limite (ε), Fuzziness(m_f).

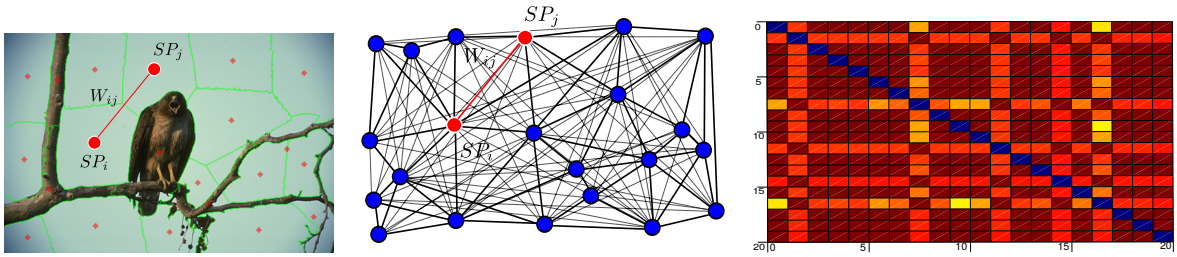
Output: Matriz prototipo $(U_{[k_{sp} \times k_f]})$.

Inicializar la matriz U con valores aleatorios entre $[0-1]$.
$p_t = \frac{2}{(m_f - 1)}$
while $max_{diff} < \varepsilon$; Detener hasta que el error sea menor al limite. **do**
  $V_i = \frac{\sum_{i=1}^{k_{sp}} U_i^{m_f} c_i}{\sum_{i=1}^{k_{sp}} U_i^{m_f}}$; Calcular los centros de los grupos.
  // Determinar los grados de pertenencia.
  for all Grupo k_f $[var : j]$ **do**
    for all Dato c_i $[var : i]$ **do**
      for all Grupo k_f $[var : r]$ **do**
        $num = \|c_i - V_j\|$; Calcular la distancia entre el dato x_i y el centro V_j.
        $den = \|c_i - V_r\|$; Calcular la distancia entre el dato x_i y el centro V_r.
        $U_{ij} = (num/den)^{p_t}$
        $sum+ = U(i,j)$
      end for

14: $val_{u_{ij}} = \frac{1}{sum}$
15: $diff = val_{U_{ij}} - U(i,j)$
16: **if** $diff > max_{diff}$ **then**
17: $max_{diff} = diff$
18: **end if**
19: **end for**
20: **end for**
21: **end while**

3.1.4. Creación de la matriz de similaridad difusa

Una imagen puede ser representada por un grafo ponderado no dirigido (ver Figura 2.2), donde los nodos representan los píxeles, y las aristas la similitud entre ellos. En la Figura 3.7 se puede observar la gráfica de similaridad de una imagen compuesta por súper píxeles.



a b c

Figura 3.7: *Construcción matriz de similaridad. a: Imagen representada por súper píxeles. b: Imagen representada por un grafo de similitud. c: Matriz de similaridad. (Realizada por los autores)*

Una buena elección de la función de pesos $w(i,j)$ influye en la calidad de la segmentación, para este algoritmo se ha elegido la función de pesos difusa propuesta en [15], la cual ha demostrado ser lo suficiente robusta para tolerar el ruido y posibles datos atípicos presentes en algunas imágenes. La función está definida como:

$$w_{ij} = \begin{cases} 1 & Si\,(x_i, x_j)\,pertenecen\,a\,la\,misma\,vecindad \\ max[u_{li}, u_{lj}],\ 1 \leq l \leq t & Si\,(x_i, x_j)\,pertenecen\,a\,la\,t\,vecindad\,próxima \\ 0 & En\,cualquier\,otro\,caso \end{cases}$$

A continuación se presenta el pseudocódigo para la construcción de la matriz de similaridad.

Algoritmo 5: Matriz de similaridad

Input: Matriz prototipo (U), Vecindario (t).

Output: Matriz similaridad ($S_{[k_{sp} \times k_{sp}]}$).

1: Obtener el número o identificador del prototipo p_{to}, al cual corresponda el mayor valor de pertenencia, para cada uno de los datos x_i de la matriz U.
2: **for all** x_i y $x_j[var : i, j]$ **do**
3: **if** $p_{to}(x_i) = p_{to}(x_j)$; x_i y x_j tienen el mismo prototipo **then**
4: $S(i, j) = 1$
5: **else if** Los prototipos $p_{to}(x_i)$ y $p_{to}(x_j)$ se encuentran dentro del vecindario t. **then**
6: $S(i, j) = max(max(U(i, j), U(i, j)))$
7: **else**
8: $S(i, j) = 0$
9: **end if**
10: **end for**

3.1.5. Cálculo de la matriz Laplaciana normalizada

El cálculo de la matriz Laplaciana parte de la construcción de la matriz diagonal $D_{[k_{sp} \times k_{sp}]}$. Se calcula la suma de cada una de las filas de S y el resultado se pone en la diagonal de D. Ya construida la matriz diagonal se procede a aplicar la ecuación de la matriz Laplaciana normalizada. Ahora se presenta el pseudocódigo de la construcción de la matriz Laplaciana.

Algoritmo 6: Matriz Laplaciana

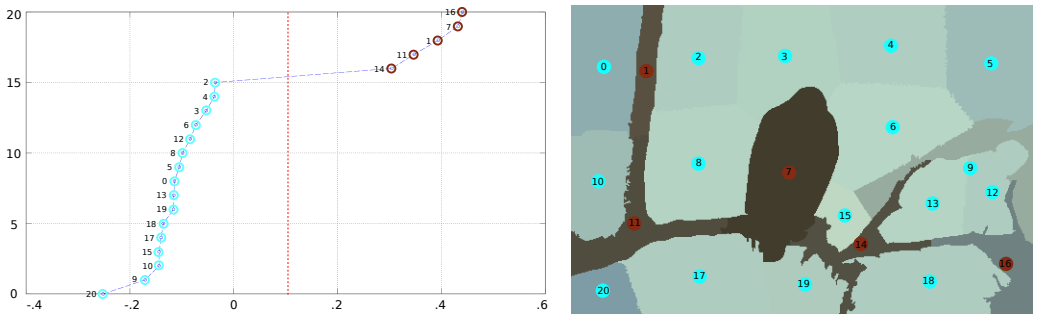
Input: Matriz Similaridad (S).

Output: Matriz Laplaciana ($L_{[k_{sp} \times k_{sp}]}$)

1: **for all** filas de S $[var : j]$ **do**
2: $D(j, j) = sum(S.row(j))$; Ubicar en la diagonal de D la sumatorio de los componente de la fila j de la matriz W.
3: **end for**
4: $L = D - S$; Calculo de la matriz Laplaciana NO normalizada.
5: $L_n = D^{-1/2} \cdot L \cdot D^{-1/2}$

3.1.6. Cálculo de los valores y vectores propios.

El cálculo de los valores y vectores propios se realiza por medio de la función *eigenvalues* de la librería *eigen*. Al finalizar los cálculos se procede a extraer los k vectores propios de menor valor; correspondientes a los k valores propios más pequeños. Para obtener una partición más adecuada se extrae únicamente el vector propio de menor valor; este vector propio se selecciona debido a su valor propio, el cual tiene que ser el de menor valor. Tener en cuenta que en el primer vector propio todos sus componentes son iguales a cero. En la Figura 3.8 se visualiza el vector propio de menor valor que determinará el mejor corte para dos grupos; este vector propio es unidimensional y contiene la misma cantidad de datos que súper píxeles en que fue pre-segmentada la imagen. Finalmente se presenta el pseudocódigo del algoritmo.



a b

Figura 3.8: *Segmentación en dos grupos; tomando el menor vector propio. a: Mejor corte de vector propio. b: Segmentación de la imagen con el corte de la Figura 3.8a. (Realizada por los autores)*

Algoritmo 7: Vectores propios

Input: Matriz Laplaciana (L_n), Número de grupos (k_f)

Output: k_f vectores propios mas pequeños ($E_{[k_{sp} \times k_f]}$).

1: Calcular los vectores propios.

2: Organizar los vectores propios de manera ascendente; de acuerdo a sus valores propios.

3: $E = eigenvalues(L)(1 \rightarrow k_f)$. Extraer los k_f vectores propios; omitir el primer vector propio.

3.1.7. Etiquetado de píxeles

El etiquetado de píxeles es el último paso, donde se construye una nueva imagen remarcando las regiones con un color específico. En la Figura 3.9 se muestra el etiquetado de la imagen; la segmentación es resultado del corte propuesto en la Figura 3.8.

a b

Figura 3.9: *Segmentación final. a: Etiquetado de píxeles. b: Superposición de imagen real con matriz de etiquetas. (Realizada por los autores)*

Algoritmo 8: Etiquetado de píxeles

Input: Matriz de vector propio (E), Número de grupos (k_f)

Output: Imagen segmentada I_{seg}

1: Lk =Etiquetado de los valores de los vectores propios con K-Means.
2: **for all** Posiciones de $L_a[var : i, j]$ **do**
3: Obtener el centroide al que pertenece el súper píxel $S_{px} = L_a(x, y)$.
4: Obtener el grupo al cual fue asignado el píxel, con respecto a su súper píxel: $Gp = L_k(S_{px})$
5: Asignar un color según el grupo al cual corresponda.
6: **end for**

Capítulo 4

Técnicas de validación.

Los métodos de validación definen el procedimiento para evaluar los resultados de los algoritmos de agrupación, de una manera cuantitativa y objetiva [51]. Donde la pregunta a resolver es: ¿qué tan bueno es el algoritmo de agrupamiento? Intuitivamente, se podría definir la efectividad de estos algoritmos, determinando si asignan al mismo grupo los puntos que son similares y a diferentes grupos los puntos disímiles [52]. Generalmente estos métodos de validación se dividen en tres grupos: subjetivos, externos e internos.

En este capítulo se estudian algunas métodos de validación propuestos en la literatura, asimismo, se define la eficiencia algorítmica mediante índices como la complejidad temporal y espacial.

4.1. Evaluación subjetiva

Este tipo de evaluación es utilizado con mucha frecuencia para determinar qué algoritmo es más idóneo frente a alguna tarea específica, en este tipo de evaluación los resultados de la segmentación son juzgados por un evaluador humano. Por lo tanto la calificación sobre una imagen puede variar según la experiencia del evaluador y en algunos casos esta variación puede llegar a ser significativa de un evaluador a otro, esto se debe principalmente a que cada evaluador genera sus propios parámetros de evaluación, como también, al orden en que sean evaluados los resultados de la segmentación. Así entonces, obtener una calificación imparcial de la eficacia de un algoritmo de segmentación es una tarea difícil.

Para llevar a cabo una buena evaluación subjetiva es necesario un gran número de imágenes. Del mismo modo, el grupo de evaluadores humanos debe ser lo suficientemente grande como para ser representativo del observador humano habitual. En consecuencia, la evaluación subjetiva es un proceso muy tedioso y consume mucho tiempo, básicamente, tales métodos no se pueden utilizar en un sistema en tiempo real [53].

4.2. Medidas de validación externas

El objetivo de esta medida es la validación, en el sentido de medir qué tan similares son entre sí un grupo de particiones distintas de un mismo conjunto de datos. Este procedimiento consiste en comparar dos particiones, la evaluada (producida por el algoritmo a ser evaluado) y una de referencia (que puede ser proporcionada por una base de datos, otro algoritmo, etc). En general, esta medida es utilizada para elegir el mejor método de agrupamiento partiendo de una estructura pre-especificada [53].

Entre los índices más importantes para la validación externa se encuentran: Recall, precisión, pureza, entropía y el Índice de Rand, los cuales serán abordados en esta sección. Pero antes, se define la matriz de confusión, ya que algunos de los índices mencionados se basan en la información de esta matriz.

Matriz de confusión

La matriz de confusión es un concepto de aprendizaje de máquina, que contiene información acerca de las clasificaciones actuales y las predichas, realizadas por un sistema de clasificación. Una matriz de confusión tiene dos dimensiones, donde las columnas contienen la información referente a la verdad, es decir, las etiquetas de las clases que se conocen de antemano. A nivel de las filas se maneja la información correspondiente al resultado del algoritmo (hipótesis) [54]. En la Tabla 4.1 se presenta la estructura de la matriz de confusión descrita anteriormente.

- **VP** (Verdadero Positivo): este término hace referencia a aquellos puntos que fueron ubicados por el algoritmo en el mismo grupo que indicaba la clase con la que se contaba de antemano.
- **FP** (Falso Positivo): hace referencia a aquellos puntos que fueron ubicados por el

algoritmo en el grupo A y que en realidad pertenecían al grupo B.

- **FN** (Falsos Negativos): hacen referencia a aquellos elementos del grupo A que fueron ubicados en un grupo diferente al que indicaba su etiqueta.
- **VN** (Verdadero Negativo): este hace referencia a aquellos elementos que fueron ubicados correctamente fuera del grupo A, es decir, aquellos elementos ajenos al grupo en cuestión y que efectivamente no corresponden a este.

Verdad / Hipótesis	Clase A	Clase B
Clase A	VP	FP
Clase B	FN	VN
$\sum$	P	N

Tabla 4.1: *Matriz de confusión. (Adaptado de [54])*

Para hacer más clara la construcción de esta matriz se expone a continuación un ejemplo: dada una imagen que está constituida por 21 píxeles $X = \{x_1, x_2, \ldots, x_{21}\}$ donde su principal atributo es el color, y se sabe que los colores existentes en la imagen son: rojo y azul. Se quiere dividir el conjunto de píxeles X en dos clases A y B respectivamente. De manera lógica se podría pensar que un buen algoritmo de agrupación sería aquel que asigna a un grupo los píxeles rojos y al otro los azules. La representación gráfica del ejemplo se visualiza en la Figura 4.1 y en base a estos datos se construirá la matriz de confusión como se muestra en la Tabla 4.1.

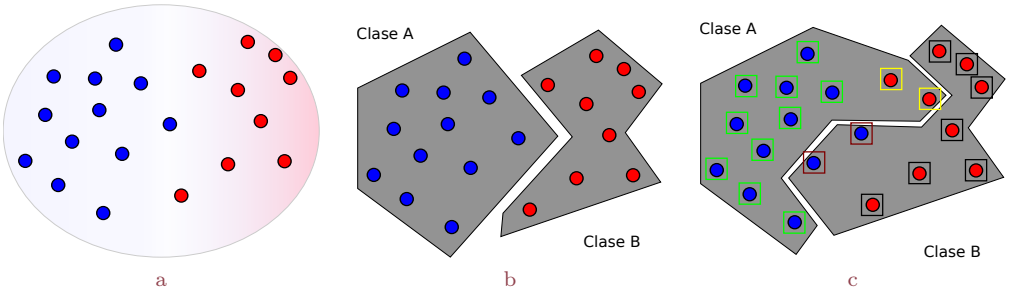


Figura 4.1: *Ejemplo de validación. a: Datos a ser agrupados. b: Agrupamiento de referencia. c: Agrupamiento realizado por el algoritmo. (Realizada por los autores)*

De los doce píxeles azules etiquetados con la clase A (ver Figura 4.1b), el algoritmo a ser evaluado acertó en un total de diez, en otras palabras el valor para **VP** en la matriz de confusión es igual a diez. Por otro lado el algoritmo agrupó en la clase A dos píxeles que en referencia corresponden a la clase B, siendo estos los falsos positivos (**FP**), en la Figura 4.1b también se aprecia que el algoritmo a ser evaluado asignó a la clase B, dos píxeles que corresponden a la clase A, los cuales se catalogan como **FN**. Finalmente de los nueve píxeles que pertenecen a la clase B el algoritmo acertó en un total de siete, valor que hace referencia a los **VN**. Con la información obtenida de la Figura 4.1 se construye la matriz de confusión como se muestra en la Tabla 4.2.

Verdad / Hipótesis	Clase A	Clase B
Clase A	10	2
Clase B	2	7

Tabla 4.2: *Matriz de confusión para el ejemplo anterior.*

En el anterior ejemplo se particionó el conjunto de datos X en dos grupos A y B, con el fin de hacer más sencilla la comprensión, sin embargo, el lector tiene que tener claro, que esta matriz puede ser construida con más grupos siguiendo la misma lógica.

4.2.1. Precisión (C), Recall (L) y medida F (F)

La precisión hace referencia a la proporción de píxeles de un grupo estimado que pertenece a la clase de referencia [54], toma esta forma:

$$C = \frac{VP}{VP + FP}$$

El valor que puede tomar esta expresión está entre 0 y 1, este valor aumenta a medida que disminuye el número de falsos positivos.

También derivado de los datos obtenidos de la matriz de confusión el índice Recall, que define la proporción de la clasificación correcta (Verdaderos Positivos) de los casos que son realmente positivos. La cual se define como:

$$L = \frac{VP}{VP + FN} = \frac{VP}{P}$$

Cuando se obtiene el mayor valor posible 1, significa que los píxeles dentro del grupo a evaluar coinciden con las etiquetas proporcionadas por la referencia, mientras que un valor para Recall igual a cero indica que el algoritmo tiene problemas para clasificar de forma correcta los datos suministrados.

La medida F representa la media armónica entre la precisión y Recall, definida como:

$$F = \frac{1 + \alpha}{\frac{1}{C} + \frac{1}{L}}$$

Para diferentes valores de α se tiene :

$$\begin{cases} \alpha = 1 & Media\ armónica \\ \alpha \to [0, 1] & preferencia\ por\ precisión \\ \alpha > 1 & preferencia\ por\ recall \end{cases}$$

4.2.2. Entropía (E) y Pureza (P)

Sea C el conjunto de clases en la base de datos X, $C = (c_1, c_2, \ldots, c_K)$. El algoritmo de agrupamiento produce k grupos, que particionan a X en k distintos grupos $D = (d_1, d_2, \ldots, d_k)$. La entropía de un grupo refleja cómo los píxeles de los k grupos se distribuyen dentro de cada grupo; la medida de calidad global se calcula promediando la entropía de todas las agrupaciones [54, 55]. Para cada grupo se asume la entropía como:

$$E(d_i) = -\sum_{j=1}^{k} Pr_i(C_j) log_2 Pri(C_j)$$

Dónde $Pri(C_j)$ es la proporción de píxeles de la clase C_j ubicados en el grupo d_i. La entropía total de todo el agrupamiento (que considera todos los grupos) es:

$$E_{Tot}(D) = \sum_{j=1}^{k} \frac{|D_i|}{|D|} E(d_i)$$

La pureza mide el hecho de que un grupo contenga solo una clase entre sus píxeles [55]. La pureza de cada grupo se calcula con:

$$P(d_i) = max_j(Pri(C_j))$$

La pureza total considerando todos los grupos es:

$$E_{Tot}(D) = \sum_{j=1}^{k} \frac{|D_i|}{|D|} P(d_i)$$

4.2.3. Índice de Rand

Propuesto por William M. Rand en [56]. Este índice determina qué tan similares son dos particiones sobre un mismo grupo de datos. Antes de hacer la definición formal de este índice se debe incluir primero el término de matriz de contingencia.

Sean $V = \{V_1, ..., V_k\}$ y $U = \{U_1, ..., U_L\}$ dos particiones sobre el mismo conjunto X de n datos. La matriz de contingencia resume el solapamiento entre los grupos de V y U (Tabla 4.3). El elemento n_{ij} de la matriz es el número de píxeles que los conjuntos V_i y U_j tienen en común, $n_{ij} = |U \bigcap V_J|$.

U \ V	U_1	...	U_L	$\sum$
V_1	n_{11}	$\cdots$	n_{1L}	a_1
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
V_k	n_{k1}	$\cdots$	n_{kL}	a_k
$\sum$	b_1	...	b_L	$\sum_{\forall_{ij}} n_{ij} = n$

Tabla 4.3: *Tabla de contingencia para la comparación de los grupos C y D. (Adaptado de [57])*

Cualquier par de píxeles x_i, x_j de un total de (n) pares distintos de X caen en alguna de las siguientes cuatro categorías:

a) x_i, x_j están en el mismo grupo en V y en D.

b) x_i, x_j están en distintos grupos en V y en D.

c) x_i, x_j están en el mismo grupo en V y en distintos grupos en U.

d) x_i, x_j están en el mismo grupo en V y en distintos grupos en U.

La cantidad de pares que caen en cada categoría se puede calcular usando la matriz de contingencia. Sean a, b, c y d los valores correspondientes a las cantidades de pares de cada categoría. Los dos primeros valores son indicadores de acuerdo entre las particiones y los dos últimos, indicadores de desacuerdo.

Con los valores ya obtenidos de a, b ,c y d se define el índice de Rand como:

$$Rand = \frac{a+b}{a+b+c+d} = \frac{a+b}{\binom{n}{2}} = \frac{\left[\binom{n}{2} + 2\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right]\right]}{\binom{n}{2}}$$

Los valores del índice están entre 0 y 1. Cuando las particiones comparadas son exactamente las mismas el valor de Rand es 1 [57].

4.3. Medidas de validación internas

Mientras que las técnicas externas comparan una imagen de referencia con los resultados arrojados por el algoritmo a evaluar, las técnicas internas, también llamadas no supervisadas, no requieren una imagen de referencia, sino que evalúan la segmentación, sobre lo bien que coinciden con un amplio conjunto de métricas ya establecidas.

Las técnicas internas son objetivas y tienen distintas ventajas. Tal vez, la principal de ellas sea el hecho de no requerir una imagen como referencia. Una imagen de referencia creada manualmente es intrínsecamente subjetiva, además, que la construcción de una base de datos como referencia es tediosa y consume mucho tiempo. Otra ventaja es que estas técnicas pueden ser usadas en tiempo real a una gama amplia de aplicaciones [53].

Las métricas en las que se basan este tipo de técnicas son por lo general la compacidad y separación, el propósito de estas medidas es determinar qué tan compactos son las agrupaciones generadas y qué tan diferente es un grupo con respecto al resto. En esta sección se tratarán algunos índices para este tipo de validación, tales como: compacidad, separación, índice de Davies-Bouldin, Coeficiente de silueta y el Índice de Dunn.

4.3.1. Compacidad

La compacidad mide qué tan cerca se encuentran los píxeles como sea posible. La suma de cuadrados intra grupo SSW (*Sum of squares within clusters*), es quizás el índice más usado para medir qué tan compacto es un grupo.

$$SSW(k) = \sum_{i=1}^{k} \sum_{x \in C_i} \|x - m_i\|^2$$

Donde k es la cantidad de grupos, m_i corresponde al centro del grupo c_i. Cuando mayor sea el valor total de SSW mayor será la relación entre los píxeles del grupo [58].

4.3.2. Separación

Las agrupaciones deben estar lo más distantes unas de las otras. Hay tres enfoques comunes que miden la distancia entre dos grupos diferentes: la distancia entre los miembros más cercanos, la distancia entre los miembros más distantes y la distancia entre los centroides de los grupos.

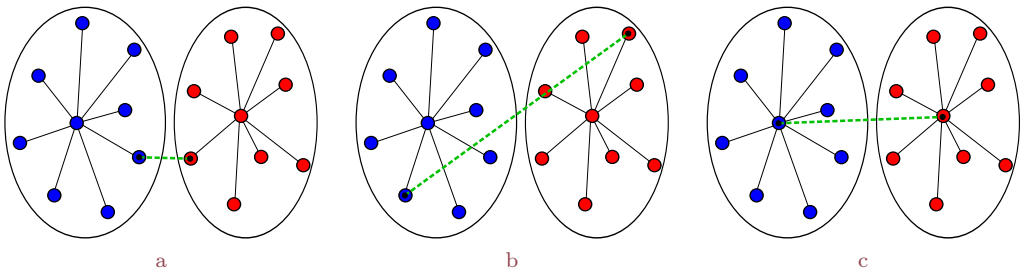


Figura 4.2: *Distancia entre grupos. a: Entre los miembros más cercanos. b: Entre los miembros más lejanos. c: Entre centroides. (Adaptada de [57])*

Uno de los índices para determinar la separaciones SSB (*Sum of squares between*).

$$SSB(k) = \sum_{i=1}^{k} |C_i| \, \|m_i - M\|^2$$

Siendo k la cantidad de grupos, C_i es el tamaño del grupo m_i y M es el centroide de los k

grupos [59]. Entre mayor sea el valor total de SSB mayor será el grado de separación con respecto a los demás grupos [58].

4.3.3. Índice de Davies-Bouldin

El índice de Davies-Bouldin se basa en la medida de similitud de los grupos c_i, c_j cuyas bases son la medida de dispersión de un grupo (c_i) y la medida de disimilitud de los grupos (d_{ij}). La medida de similitud entre agrupaciones se define como:

$$DB = \frac{1}{k} mx_{i \neq j} \left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)} \right)$$

Donde k es el número de grupos, c_x denota el centroide del grupo x, σ_x es la distancia media de todos los elementos del grupo x al centroide c_x, y $d(c_i, c_j)$ es la distancia entre los centroides c_i y c_j.

El máximo valor representa el peor caso para el grupo i. La solución óptima es aquella que tiene el índice de Davies-Bouldin más bajo [60].

4.3.4. Índice de Dunn

Muy similar al índice Davies-Bouldin, el objetivo de este índice es identificar un conjunto de grupos que sean compactos, con una varianza pequeña entre los píxeles del grupo, y que éstos estén bien separados de los píxeles de otros grupos. Un valor más alto del índice de Dunn indica un mejor rendimiento del algoritmo de agrupamiento. Por lo tanto, este índice también sirve para encontrar en número óptimo de grupos en un conjunto de datos [51].

El índice de Dunn tiene un valor entre cero e infinito, y debe ser lo más alto posible. Por lo tanto, la distancia entre los píxeles de un grupo debe ser lo más baja posible, y la distancia entre los grupos lo más alta como sea posible. La siguiente ecuación define el índice de Dunn.

$$D = \min_{1<i<n} \left\{ \min_{1<j<n; i \neq j} \left\{ \frac{d(i,j)}{\max_{1<k<n} d'(k)} \right\} \right\}$$

Donde $d(i, j)$ representa la distancia entre los grupos i y j, y $d'(k)$ mide la distancia dentro del grupo k. El lector debe tener en cuenta que no queda fijo en la definición qué medida entre grupos se debe utilizar [61, 62] ver Figura 4.2.

4.4. Eficiencia algorítmica

El término eficiencia algorítmica es usado para describir aquellas propiedades de los algoritmos que están relacionadas con la cantidad de recursos utilizados por el algoritmo. Las medidas más usadas para determinar la eficiencia de un algoritmo son: complejidad temporal y espacial. Teniendo como objetivo lograr la mayor eficiencia, lo que se busca es minimizar el uso de recursos por parte del algoritmo, sin embargo, es muy común escoger una de estas medidas como prioridad.

4.4.1. Complejidad temporal

El tiempo de ejecución de un algoritmo va a depender de diversos factores como son: los datos suministrados a la entrada, la calidad del código generado por el compilador para crear el programa objeto, la naturaleza y rapidez de las instrucciones de máquina del procesador que ejecute el programa, y la complejidad intrínseca del algoritmo [63]. El tiempo de CPU y el de pared (*wall time*) son dos métricas para medir el tiempo de ejecución de un algoritmo donde el *wall time* mide el tiempo de la forma en que es percibido por el humano, es decir el tiempo transcurrido desde que inició el algoritmo hasta cuando este finalizó su ejecución, el problema de este tipo de medida es que tiene en cuenta el tiempo que la máquina ha dedicado a otras tareas, en cambio el tiempo CPU mide el tiempo dedicado específicamente al algoritmo en cuestión [64].

4.4.2. Complejidad espacial

La complejidad espacial es la cantidad de memoria necesaria para realizar las operaciones computacionales que requiere un algoritmo. Incluye todas las variables, tanto globales como locales, como también las estructuras de datos. La complejidad espacial es más difícil de calcular que la complejidad temporal porque no todas estas variables y estructuras de datos pueden ser necesarias al mismo tiempo. Hay que tener en cuenta que las variables

existen y ocupan espacio de memoria todo el tiempo. Las variables locales (y la información guardada en la pila) sólo existirán durante la invocación de la función. En resumen si se tiene un número constante de variables, también la complejidad espacial será constante [65].

Finalmente, después de realizar el estudio de las métricas de evaluación, se procede a escoger las más representativas dentro de cada categoría, teniendo en cuenta el enfoque del algoritmo a ser evaluado. Para este trabajo se han escogido las siguientes métricas ya que proporcionan criterios acordes con los objetivos a ser alcanzados en el mismo.

- Precisión (C)
- Recall (L)
- Medida F (F)
- Compacidad (SSW)
- Separación (SSB)
- Complejidad Temporal

Capítulo 5

Experimentación y Evaluación.

La experimentación y evaluación es el paso final que permite determinar el rendimiento del algoritmo propuesto, siendo necesario realizar el análisis con respecto a otro algoritmo teniendo en cuenta algunos indicadores como Complejidad Temporal, Precisión (C), Recall (L), Medida F (F), Compacidad (SSW) y Separación (SSB); presentados en el anterior capítulo. Con este análisis se podrán vislumbrar las diferencias principales entre estos métodos de segmentación de imágenes y el algoritmo SFSC.

5.1. Base de datos

Para la experimentación del algoritmo se ha utilizado la base de datos BSD300 (*Berkeley Segmentation Dataset 300*) [68], consta de 300 imágenes a color, y en escala de grises. Las imágenes se dividen en un conjunto de 200 imágenes de entrenamiento y un conjunto de 100 imágenes de prueba. Además, cuenta con la posibilidad de descargar la segmentación de las imágenes realizada por personas.

La base de datos consta de una variedad de imágenes que presentan diferentes retos para el algoritmo. Las imágenes van desde ambientes naturales hasta rostros de personas, con lo cual se tienen imágenes texturizadas, donde hay diferenciación simple entre objetos e imágenes con complejas estructuras. Los resultados de 100 imágenes segmentadas son agregadas en el anexo B.

5.2. Algoritmos

Ahora se realiza una pequeña descripción de los algoritmos a evaluar, cabe aclarar que estos presentan similitud en la esencia de agrupamiento espectral y, además varios de ellos son explicados en los capítulos 1 y 2. Los algoritmos con los cuales se va a desarrollar la evaluación son:

*Shi*Malik (**SM**): Se trata la segmentación de imágenes como un problema de partición de grafos, el mayor aporte es el criterio global de corte normalizado, para segmentar el grafo; esto consiste en dejar bien definido los puntos de corte en los vectores propios seleccionados, principalmente para la bipartición. El criterio de corte normalizado mide tanto la disimilitud total entre los diferentes grupos como la similitud total dentro de los grupos. Se muestra que es una técnica eficiente [36]. (D)

Nyström (**NW**): Este método permite dar solución al algoritmo de agrupamiento espectral extrapolando la solución de los valores y vectores propios (espectro) utilizando un pequeño número de muestras de la matriz de características, seguidamente se selecciona el vector propio de menor valor, este será agrupado según el número de segmentos deseado [66]. (I)

Spectral (**SC**): Este método calcula la matriz Laplaciana con su propuesta, la matriz Laplaciana normalizada de Jordan y Weiss. Además se diferencia con los otros algoritmos por el uso de k vectores propios mas pequeños, en vez de uno solo para segmentar [45]. (I)

K-Means **(KM):** El algoritmo es de fácil implementación, y permite obtener grupos rápidos (*Adaptado con operación morfológica de dilatación)* (I)

Naotoshi (**NS**): El algoritmo consta de cuatro pasos. Primero, se construye la matriz simétrica de similaridad. Segundo, se resuelve el sistema generalizado de valores y vectores propios. Tercero, se realiza el corte del vector propio de menor valor, en el caso ideal los valores del vector propio presentan una separación bien definida. Finalmente se realiza la bipartición recursivamente [67]. (D)

BaiaCaoa (**NC**): La segmentación se realiza en dos etapas. Primero, se pre-segmenta la imagen por el método de SLIC (*Simple linear Iterative Clustering*) modificado,

donde se extraen las regiones "globales" que contengan características similares, y segundo, se realiza el agrupamiento de las regiones pre-segmentadas haciendo uso del método de Nyström [13]. (I)

Los algoritmos denotados con D fueron descargados de la página de los desarrolladores en su versión para matlab®; y los denotados con la letra I son algoritmos implementados, para **Nyström** se implementa la función de extrapolación de valores y vectores propios tomándola del artículo, con la implementación de **Nyström** se desarrolla **NC** agregando súper píxeles; descargando la función de SLIC desarrollada por Achanta. K-Means es desarrollado y modificado con una operación morfológica de dilatación; los anteriores algoritmos son implementados en matlab® 2014a. **Spectral** como **SFSC** son implementados bajo el lenguaje de programación C++, teniendo en cuenta los criterios de diseño expuestos en el capitulo 3.

5.3. Complejidad temporal

Para el primer experimento del algoritmo se evaluará el desempeño de la complejidad temporal, se tiene en cuenta que los algoritmos a comparar con SFSC (**SF**) se encuentran desarrollados sobre la plataforma Matlab®, lo cual da una ventaja; para el tratamiento de matrices y a la solución del problema de valores propios, pero con la utilización de Intel® MKL para la librería *eigen* se tiene un rendimiento "comparable" con los demás algoritmos.

Otras características de los algoritmos para tener en cuenta en la evaluación[1]:

1. Todos los algoritmos trabajan sobre el espacio de colores CIELab, además se incluye la ubicación espacial como descriptores.

2. El eje x de la Figura 5.2 representa las dimensiones de las imágenes seleccionadas para la evaluación de complejidad temporal.

3. Los datos de complejidad temporal son tomados sobre cuatro imágenes donde se presentó menor rendimiento, se parte del criterio de tomar los datos en el peor de los escenarios. Las imágenes son redimensionadas a tamaños de $(m \times m)$ que van desde los 10×10 hasta 1000×1000 píxeles; y se segmentan en dos y tres grupos.

[1]Las características 2 y 3 son válidas para la siguiente subsección.

Figura 5.1: *Imágenes seleccionadas para complejidad temporal.*

4. La ejecución de los algoritmos se realizó sobre un PC: Intel® Core™ i3 CPU: M-370 Frecuencia: 2.40GHz y RAM: 2GB.

En la Figura 5.2 se presentan los resultados de complejidad temporal medida en tiempo de CPU, esto se debe a que es dedicado específicamente al algoritmo. El tiempo se obtiene de la media de los tiempos empleados para segmentar las cuatro imágenes, como de la segmentación en la cantidad de grupos expuestos en el ítem 3.

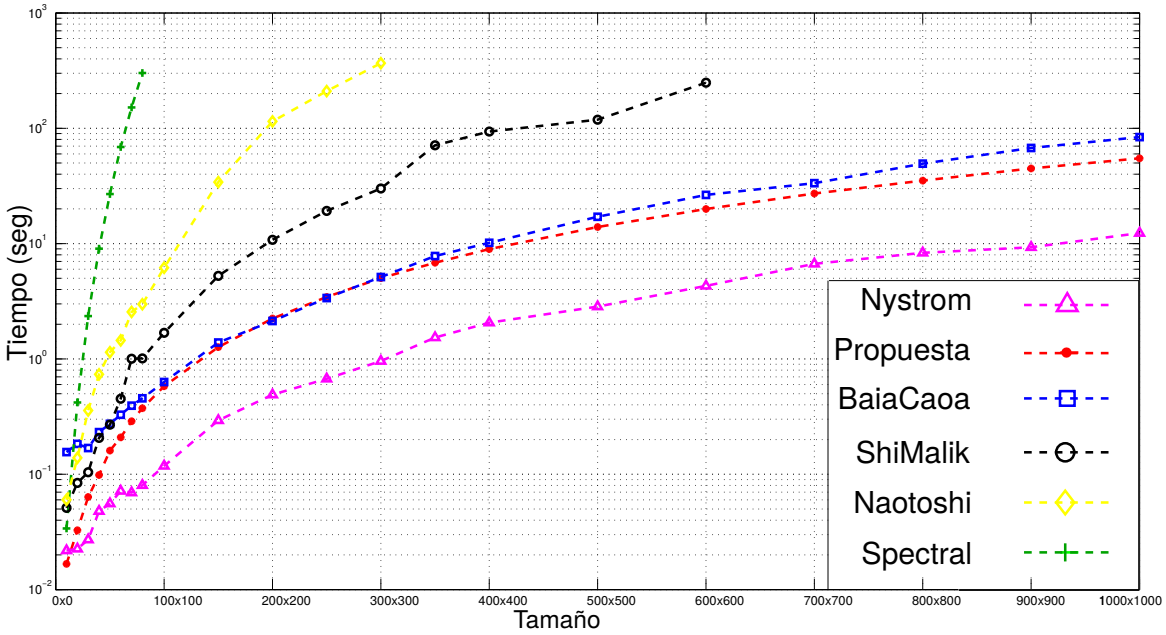


Figura 5.2: *Gráfica de tiempo; se gráfica la media de los valores de tiempo obtenidos para la segmentación de las 4 imágenes. Ejecución de los algoritmos SFSC (SF), Nyström (NW) [66], BaiaCaoa (NC) [13], Spectral Clustering-Jordan and Weiss (SC) [45], Normalized Cuts-Shi and Malik (SM) [36], Normalized Cuts-Naotoshi Seo (NS) [67]. (Realizada por los autores)*

De la Figura 5.2 se tiene la mejor eficiencia por parte del algoritmo **NW**, esto se debe principalmente a que el algoritmo permite solucionar el problema de valores y vectores

propios con una baja cantidad de elementos de la matriz de características; por este motivo se ahorra el cálculo de la matriz Laplaciana, como la obtención de los valores y vectores propios con todos los datos. Aunque **SF** disminuye la cantidad de datos a procesar para el cálculo de la matriz Laplaciana, no se ha dejado de aplicar el agrupamiento espectral tradicional, además la pre-segmentación y la construcción de la matriz prototipo fuzzy aumentan en pequeñas cantidades el tiempo de ejecución.

NC y **SF** tienen un rendimiento comparable debido a que presentan una etapa de pre-segmentación basada en SLIC, la cual es el factor que determina un mejor rendimiento para **NW**, se puede decir esto porque **NC** utiliza Nyström como método de solución de los vectores propios y, a pesar de esto no consigue superar el rendimiento de **SF**. Cabe decir que **NW** para algunos casos sufre de indeterminación en la solución del problema de valores y vectores propios; como resultado se tiene la interrupción del cálculo de la segmentación. **NC** y **SF** presentan menor rendimiento cuando el número de súper píxeles es muy pequeño, esto se debe a que la imagen "quedaría" segmentada por el algoritmo K-Means tradicional; el número de grupos a segmentar ciertamente estaría muy cercano al número de súper píxeles seleccionados, por tanto, la diferencia entre la pre-segmentación y la segmentación final sería casi nula.

Los algoritmos que se ubicaron en los últimos puestos se debe a que realizan la segmentación con todos los píxeles de la imagen, cabe aclarar que realizan el cálculo de la matriz Laplaciana como el de los valores y vectores propios con todos los píxeles (**SC**, **SM**, **NS**), además realizan una estructura en árbol; lo que quiere decir que se ejecuta iterativamente hasta realizar los mejores cortes, por ejemplo se divide la imagen original en dos regiones, seguidamente las dos regiones son tratadas como "imágenes" independientes pero manteniendo la jerarquía, donde se segmenta estas regiones en otras aplicando de nuevo el algoritmo en cada región y así se sigue hasta determinar las regiones deseadas (**SM**, **NS**); y además **NS** busca obtener el mejor número de grupos; aquí se aplica otro algoritmo precedente al agrupamiento espectral que determina según la estructura de la imagen el mejor número de grupos, este valor funciona como parámetro de entrada para el agrupamiento espectral. Para $m > 600$ a estos tres algoritmos se les deja de ejecutar porque muestran una tendencia excesiva a demorar, esto se debe a que tienen que procesar una gran cantidad de datos.

5.4. Evaluación externa

Para obtener los resultados de las medidas externas se tiene en cuenta los cuatro algoritmos que mejor desempeño tuvieron en el experimento anterior **SF**, **NW**, **NC** y **SM**, además se incluye K-Means (**KM**), el algoritmo es incluido debido a que dos de los algoritmos **NC** y **SF** presentan cierta semejanza con este, por el algoritmo SLIC; se aclara que es para "observar" el desempeño de este algoritmo con respecto a los demás. Los algoritmos no nombrados de la anterior sección son retirados porque no cumplen los requerimientos para ser comparados con **SF**; principalmente porque la segmentación de las imágenes de la base de datos emplearía un tiempo excesivo.

Para este segundo experimento se tienen en cuenta algunos criterios expresados en la sección 5.3, además se tienen en cuenta otros criterios como:

1. Modificación de las segmentaciones realizadas por las personas de la base de datos BSD300 para obtener grupos globales; ya que la base de datos presenta grupos locales, esto quiere decir que regiones en un contexto global son símiles con otras, pero la base de datos las representa como regiones distintas porque están separadas espacialmente.

2. K-Means es adaptado con una función de dilatación binaria (la dilatación es una operación morfológica que amplia y realza zonas, esta función representa un crecimiento progresivo de un conjunto X al pasar un elemento estructurante dentro del conjunto, este no se modifica), lo cual evita tener una excesiva cantidad de regiones aisladas.

En la Tabla 5.1 se presentan los valores de Precisión(C), Recall(L) y Media armónica(F) para los algoritmos **SF**, **KM**, **NC**, **NW** y **SM**. En la tabla se resaltan los valores de la Media Armónica para quien ocupa el primer lugar (verde) y quien ocupa el segundo (magenta); la selección de la media armónica como punto de análisis se debe a la falta de preferencia de una métrica sobre la otra.

I \ A	SF			KM			NC			NW			SM		
	Medida														
	C	L	F	C	L	F	C	L	F	C	L	F	C	L	F
101085	8897	9060	8978	9141	9079	9110	8871	9033	8952	6745	6723	6734	5597	7813	6523
14037	8934	9205	9068	7716	7713	7715	5829	5055	5415	5694	5553	5623	7022	7034	7028
143090	5810	4833	5277	6812	6669	6740	6384	7213	6773	5934	5462	5688	7433	7354	7394
147091	9769	9817	9793	7576	8464	7995	8656	9102	8874	8882	9145	9012	5840	5939	5889
208001	8041	6473	7173	8220	6514	7268	7497	6069	6727	7471	6102	6718	7785	6226	6919
295087	8762	9149	8951	7225	7454	7338	7811	7922	7866	9798	9733	9766	5327	5321	5324
296007	6807	6464	6631	5867	5653	5758	7113	7054	7084	5298	5845	5558	4832	3717	4202
296059	8160	7874	8014	5917	5066	5458	4861	5155	5004	7629	7286	7454	5120	5922	5492
304034	8110	7072	7556	7719	6896	7285	7212	6166	6648	7788	6761	7238	6482	5851	6151
3096	9101	9514	9303	6280	9772	7647	7555	5580	6419	7673	5607	6479	5495	5112	5297
41033	8780	9070	8923	7576	8515	8019	8101	8089	8095	6680	6673	6677	5258	5261	5260
42049	9386	8827	9098	9645	9213	9424	9017	8120	8545	8774	7595	8142	6735	6114	6409
69015	6819	6677	6747	4348	3456	3852	3680	4929	4214	5143	4268	4665	5396	4796	5079
86016	7417	7880	7641	7414	9007	8133	5844	5740	5792	6391	6392	6392	5837	5805	5821

Tabla 5.1: *Medidas: Precisión (C), Recall (L) y Media armónica F (F) para las imágenes presentadas en la Figura 5.3. (Los datos son multiplicados por un factor de $\times 10^4$. Primer puesto: Verde; Segundo puesto: Magenta).*

De la Tabla 5.1 se puede observar como el algoritmos **SF** presenta la mayor cantidad de aciertos para la media entre Precisión y Recall (se toma como referencia este valor debido a que el segmentador no tiene una aplicación definida; por tanto no hay decisión en favorecer alguna de las medidas). **SF** segmentó 7 de 14 imágenes como mejor segmentación y 6 de 7 como segunda mejor segmentación con un error medio de 3.855% con referencia a la mejor segmentación. La gran cantidad de acierto que presentó **KM** se debe a la operación morfológica de dilatación, aplicando esta operación más píxeles pueden ser asignados a un grupo, por tanto la cantidad de aciertos en la matriz de confusión aumenta.

De la Figura 5.3 se observa como el segmentador **NW** tiene una gran ventaja en encontrar segmentos pequeños para ser asignados a un segmento más lógico, a **NC** y **SF** se les dificulta encontrar este tipo de regiones debido a la implementación de SLIC como pre-segmentación. Por otra parte, el algoritmo **SM** no tuvo un gran desempeño debido a una "baja" ponderación de la distancia espacial entre píxeles, como resultado se tiene y se observa de la Figura 5.3 la construcción de diagramas de Voronoi, cabe destacar que el algoritmo mejora su rendimiento cuando se aumenta la cantidad de segmentaciones a realizar.

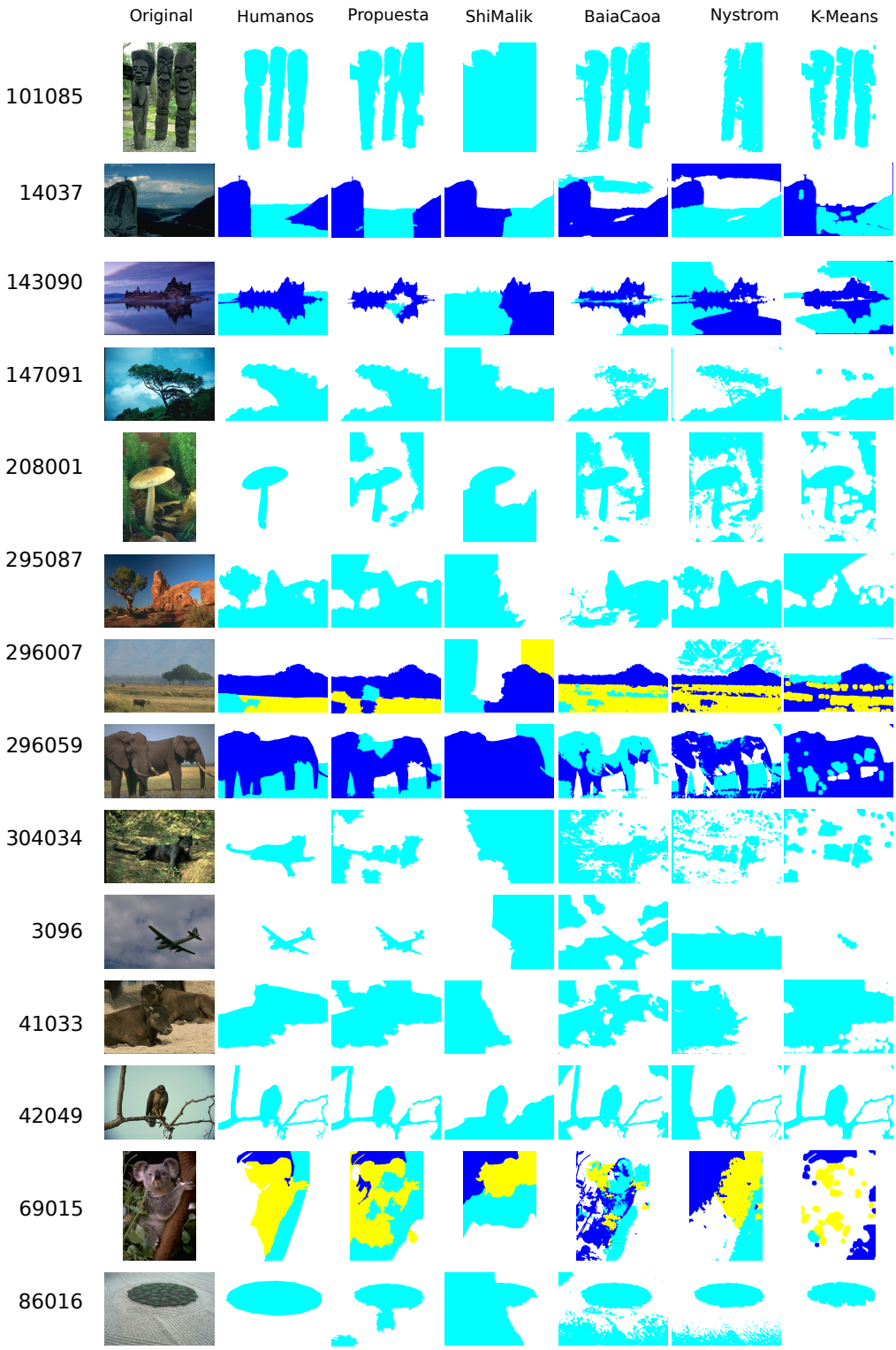


Figura 5.3: *Segmentación de imágenes. De izquierda a derecha las segmentaciones: Original, Humana, K-Means (KM), Normalized Cuts (SM), SLIC-Nystrom (NC), Nystrom (NW), SFSC(SF). (La cantidad de colores que aparece en la segmentación corresponde a la cantidad de grupos seleccionados a segmentar). (Realizada por los autores)*

5.5. Evaluación Interna

Para este experimento se tienen en cuenta las segmentaciones realizadas en la sección 5.4 y características definidas en las anteriores secciones. Las métricas de evaluación que se tienen en cuenta son *SSW* y *SSB* la cuales determinan compacidad y separación de manera individual. En la Tabla 5.2 se presentan los datos obtenidos de las métricas. El análisis se basa en que un valor bajo de compacidad del algoritmo es mejor y por el contrario, entre más grande sea la separación es mejor el algoritmo. En la Tabla 5.2 se han remarcado los mejores valores obtenidos por los algoritmos, el color verde representa el mejor resultado para la métrica y en color magenta se tiene el segundo valor; se tiene en cuenta esta consideración debido a que hay una diferencia baja entre los algoritmos para las métricas presentadas.

I \ A	SF		KM		NC		NW		SM	
	Medida									
	SSW	SSB	SSW	SSB	SSW	SSB	SSW	SSB	SSW	SSB
101085	1.3329e7	5.2518e4	1.3352e7	2.9824e4	1.3330e7	5.2094e4	9.6450e2	1.3376e7	1.3376e7	5.8662e3
14037	5.1517e6	5.3681e4	5.1115e6	9.9578e4	5.1649e6	4.6237e4	5.1822e6	2.8924e4	5.1388e6	7.2334e4
143090	2.8615e6	6.3286e3	2.8653e6	2.5582e3	2.8638e6	4.0205e3	2.8666e6	1.2449e3	2.8654e6	2.3820e3
147091	2.5402e7	6.6448e5	2.6045e7	2.1706e4	2.5665e7	4.0139e5	2.5384e7	6.8322e5	2.5891e7	1.7612e5
208001	1.2269e5	4.0872e0	1.2272e5	2.3478e0	1.2271e5	1.4593e0	1.2269e5	1.2753e1	1.2270e5	2.7210e0
295087	2.1583e6	1.5385e3	2.1584e6	1.5381e3	2.1590e6	1.45015e3	2.1574e6	2.4268e3	2.1597e6	1.6114e2
296007	6.9750e	7.0036e	7.2249e	7.2419e	6.3278e	6.3491e	1.6931e	1.6764e	1.9989e	1.9852e
296059	1.3130e	1.3154e	1.0779e	1.0791e	7.7286e	7.7648e	6.1749e	6.1999e	1.0768e	1.0730e
304034	1.8075e7	3.3100e5	1.8227e7	1.7935e5	1.8343e7	6.3039e4	1.8082e7	3.2400e5	1.8366e7	3.9697e4
3096	1.7699e6	3.1192e4	1.7571e6	4.4004e4	1.7991e6	2.0062e3	1.7991e6	1.9723e3	1.8010e6	7.2504e1
41033	1.1080e6	3.9195e2	1.1081e6	2.6635e2	1.1078e6	6.0218e2	1.1078e6	6.1768e2	1.1082e6	1.8389e2
42049	7.0355e5	1.1448e3	7.0347e5	1.2282e3	7.0379e2	9.0542e2	7.0404e5	6.6012e2	7.0419e5	5.0470e2
69015	6.4669e7	1.4335e6	6.5465e7	6.3797e5	6.3156e7	2.9467e6	6.4062e7	2.0408e6	6.5548e7	5.5500e5
86016	8.3802e	8.3491e	2.3936e	2.3872e	1.9991e	2.0055e	3.0905e	3.0305e	4.5648e	4.6483e

Tabla 5.2: *Medidas: Compacidad o cohesión (SSW) y Separación (SSB) para los algoritmos SFSC, K-Means (KM), Slic-Nystrom (NC), Nystrom (NW) y Normalized Cuts (SM) para las imágenes presentadas en la Figura 5.3.*

De la Tabla 5.2 se tienen mejores resultados para compacidad con el algoritmo **NW**, presenta 5 de 14 valores en primer lugar, y 4 de 9 valores en segunda posición. Seguidamente el algoritmo **SF** presenta buenos valores de compacidad con 4 de 14 en primer lugar, y 5

de 10 en segunda posición. Para la separación entre grupos los algoritmo **SF**, **KM** y **NW** son los que mejor separan los diferentes grupos con 4 aciertos de 14 para cada uno, **SF** hace 6 de 10 aciertos en segunda posición, dando buenos resultados para estas métricas. Una observación general de la tabla es la presencia de valores homogéneos entre todos los algoritmos, se puede inferir que estos no difieren mucho en crear grupos compactos y a la vez separados.

Un buen agrupamiento deberá ser aquel que contenga cada uno de los grupos individuales homogéneos y entre todos los grupos tan heterogéneos como sea posible. Para este análisis se van a tener en cuenta las dos métricas de forma conjunta, se realiza el cálculo de los datos presentados en la Figura 5.4 con las imágenes de la Figura 5.3 y con los valores presentados en la Tabla 5.2, para cada algoritmo con la ecuación $log(\frac{SSW}{SSB} + 1)$. El logaritmo se aplica para acotar el resultado, además se suma 1 para evitar algunos resultados negativos, para efectos del análisis estas modificaciones no generan inconvenientes; porque son usadas para generar una mejor gráfica. Ahora, un valor pequeño de $\frac{intra-Cluster}{inter-Cluster} = \frac{SSW}{SSB}$ representa homogeneidad dentro de los grupos y heterogeneidad entre los grupos.

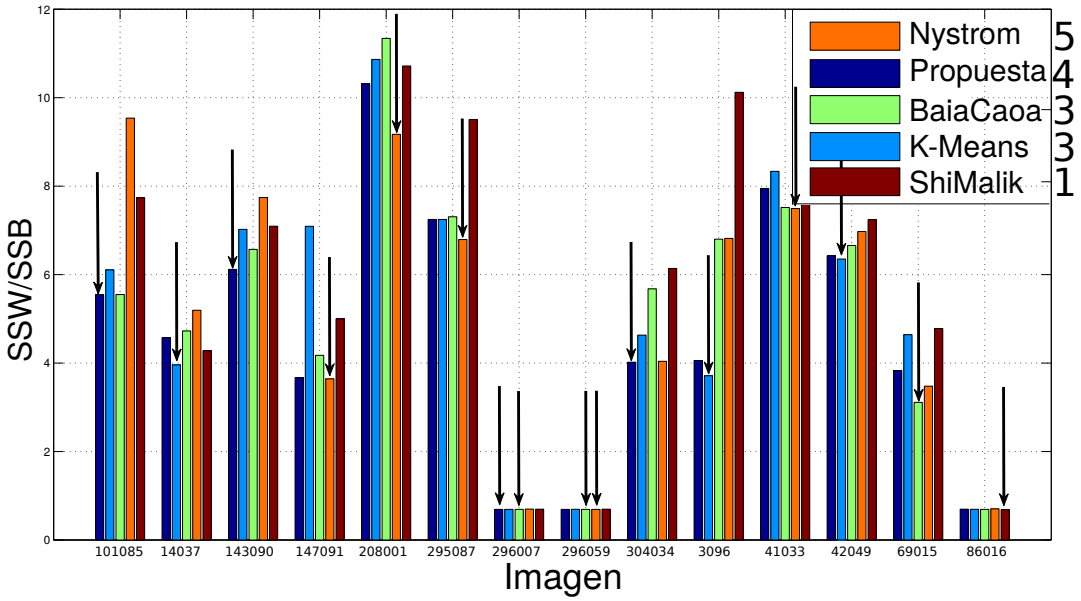


Figura 5.4: *Cuantificación de la dispersión de los puntos a nivel inter/intra-cluster para las imágenes de la Figura 5.3.*

Analizando la métrica conjunta $\frac{SSW}{SSB}$ para determinar qué algoritmo presenta mejores resultados se tiene la gráfica de la Figura 5.4. Las flechas dentro de la figura representan el mejor resultado, cabe aclarar que la doble flecha representa un mismo valor de la métrica

para dos algoritmos. Se observan mejores resultados para el algoritmo **NW** con 5 de 14 aciertos (4, 5, 6, 8, 11), además muestra buenos resultados para las demás imágenes segmentadas. La mayor cantidad de aciertos se encuentran entre los algoritmos **NW**, **SF** y **KM** con once aciertos. Los algoritmos analizados presentan cierta semejanza en los resultados presentados.

Capítulo 6

Conclusiones y Trabajos futuros

En este capítulo se resumen las principales conclusiones a las que se ha llegado después de explorar y comprender los temas que conforman esta monografía. Asimismo se plantean las futuras vías de investigación para la mejora del mismo.

6.1. Conclusiones

El objetivo que ha guiado la realización de la presente monografía ha sido el desarrollo de una técnica de segmentación de imágenes que permite resolver dos problemas principales. Primero, reducir los tiempos de ejecución y el segundo principio lograr una segmentación en regiones coherentes sobre una imagen dada, por tanto consideramos que los objetivos planteados se han alcanzado satisfactoriamente por la propuesta.

Vemos cómo el algoritmo desarrollado ha reducido considerablemente el tiempo de respuesta en comparación al algoritmo clásico (**SC**), la razón principal es la introducción de la técnica de súper píxeles en la etapa de pre-segmentación, la cual reduce los datos a ser procesados por agrupamiento espectral, y en consecuencia reduce el tiempo de ejecución de **SFSC**. Debido a que un algoritmo clásico trabaja con todos los píxeles de la imagen, este se ve afectado por imágenes de más de 2500 píxeles, a diferencia del algoritmo propuesto el cual mostró un buen desempeño a la hora de segmentar imágenes conformadas por $100e^3$ píxeles. Cabe destacar que la eficiencia del algoritmo se ve afectada cuando el número de súper píxeles se acerca al número de píxeles en la imagen.

La calidad de la segmentación también fue superada por el algoritmo propuesto, esto gracias al uso de la función de similaridad difusa, ya que con esta medida se tiene una gran separación entre grupos y mayor cohesión intra grupos. La ventaja de construir la matriz de similaridad bajo la función difusa, se ve reflejada en el criterio de partición de la imagen, puesto que permite a la matriz Laplaciana y posteriores vectores propios dar puntos de corte más visibles para el algoritmo K-Means.

Finalmente, la selección de un único vector propio para la segmentación trae mejores resultados que seleccionar k vectores propios, esto se debe a que el vector propio presenta una mejor distinción entre grupos, con lo cual K-Means puede agrupar fácilmente los datos del vector. Tomar los k vectores propios presenta un problema k dimensional de agrupamiento para K-Means, por tanto conlleva a un bajo desempeño del algoritmo propuesto, además, el uso de un semillado aleatorio para K-Means genera inestabilidad en el agrupamiento de los datos, causando variaciones en la segmentación final, el problema se hace más visible cuando se ingresa una gran cantidad de datos a agrupar.

6.2. Trabajos Futuros

Como continuación del trabajo realizado, y en margen de los objetivos fijados, quedan abiertas algunas líneas de investigación que consideramos interesantes.

Se plantea extender la caracterización de los píxeles a otras características, por ejemplo descriptores de texturas, en particular, planteamos introducir el uso de descriptores difusos que recojan la imprecisión en la caracterización de un píxel. En relación con este punto, se plantea asimismo la definición de medidas de semejanza y conectividad entre píxeles que tomen en consideración las nuevas caracterizaciones y su aplicación a la segmentación de imágenes, de esta modo sería posible realizar segmentaciones sobre la base. Por ejemplo, de las texturas presentes en la imagen, con respecto a la cual la segmentación basada en homogeneidad en color sería, de hecho, un caso particular.

El número de grupos como el de súper píxeles en el algoritmo actual son ingresados de forma manual. Deberían explorarse mecanismos que permitan la obtención automática y óptima de estos parámetros.

Finalmente, no se ha explorado lo suficiente las configuraciones de K-Means para mejorar la segmentación, por tanto se podrían evaluar parámetros que mejoren su comportamiento o incluso evaluar la implementación de otro método de agrupamiento.

Anexos

Capítulo A

Manual de usuario

El presente manual de procedimientos tiene como propósito guiar de forma específica la óptima operación y desarrollo de las diferentes actividades para la instalación de complementos y uso adecuado de la aplicación desarrollada, así como servir de instrumento de apoyo y mejora de la aplicación. El manual se divide en dos secciones.

En la primera sección se describe la correcta instalación y configuración de las librerías. En la segunda sección se describe la aplicación, y el uso de la interfaz gráfica de usuario.

Es importante considerar que el proceso de instalación de las librerías y programas adicionales; se realiza en el sistema operativo basado en Linux (Debían), por tanto se espera que el lector tenga conocimiento sobre el entorno Linux.

A.1. Instalación de librerías

En esta sección se explica la instalación y configuración de las librerías. Puntualmente se da una descripción general de cada librería, las páginas de descarga, los pasos para la descarga y los comandos de consola con el fin de concluir con una exitosa instalación.

A.1.1. Intel® Math Kernel Library (Intel® MKL)

Es una biblioteca de rutinas matemáticas optimizadas para ciencia, ingeniería y aplicaciones financieras. Las funciones matemáticas principales incluyen las bibliotecas BLAS,

LAPACK y ScaLAPACK, solucionadores dispersos, transformadas rápidas de Fourier y matemática de vectores. Las rutinas en MKL están optimizadas a mano, explotando los procesadores multicore y de many core actuales, unidades vectoriales de mayor longitud de palabra y otras características relacionadas con la arquitectura del procesador. La biblioteca MKL ayuda a los desarrolladores a concentrarse en el dominio de la aplicación, ahorrando tiempo y costes de desarrollo, depuración y mantenimiento. Los pasos que se deben seguir para su instalación son:

1. Ir el centro de registro de productos de software Intel® y registre el producto proporcionando su e-mail y número de serie, si ha realizado la compra del software. Sin embargo, Intel permite la descarga de forma gratuita aceptando un acuerdo de licencia de uso no comercial del software. Para ello, debe ingresar a su página de internet[1] , seleccionar el software, introducir algunos datos personales (nombre y apellido del usuario, correo electrónico, tipo de usuario (Individual), etc ...), la página se redireccionará automáticamente y mostrara un número de serie con el cual se puede realizar la instalación del programa. Debajo del número de licencia, aparecerá un enlace para proceder a la descarga. Antes de pulsar, conviene copiar o apuntar el número de serie, aunque la página enviara un correo electrónico a la cuenta especificada con la clave generada.

2. Si la cuenta de inicio de sesión no tiene acceso administrativo y el directorio de instalación requiere privilegios de escritura administrativa del sistema (por ejemplo, */opt/intel*), es posible que necesite la ayuda del administrador del sistema para instalar los paquetes.

	Especificaciones Técnicas
Hardware Requerido	Intel® y compatibles, incluyendo pero no limitado a: Procesadores Intel® Xeon®, la familia de procesadores Intel® Core ™ y la familia de procesadores Intel® Atom ™.
Sistema Operativo	Utiliza la misma API para el desarrollo de aplicaciones en varios sistemas operativos: Windows *, Linux * y OS X *.

[1] https://software.intel.com/en-us/intel-mkl

Herramientas de desarrollo y Ambientes	Microsoft Visual Studio* (Windows*) Eclipse (Linux* and OS X*)
Lenguajes de Programación	Nativamente se realiza el desarrollo en C, C++ y Fortran. Compatible con Java*, C#, Python * y otros lenguajes.

Instalación MKL

El proceso de instalación de MKL, una vez descargado, es el siguiente:

1. Ejecutar el script **install_GUI.sh** ubicado en el nivel superior del archivo de instalación como se muestra en la Figura A.1. El panel izquierdo de la ventana de instalación proporciona un informe del progreso de instalación, los pasos que se ejecutan en la instalación son:

   - Acuerdo de licencia
   - Requisitos previos
   - Activación de licencia
   - Programa Intel® de mejora de software.
   - Opciones
   - Instalación
   - Completar

   Leer y aceptar los términos del Acuerdo de Licencia de Usuario Final (CLUF), hacer clic en el botón **Guardar**, de ese modo el texto del acuerdo de licencia se almacena en el sistema, hacer clic en el botón **Siguiente**.

```
root@localhost:~# tar xzvf l_mkl_2017.0.098.tgz
root@localhost:~# cd l_mkl_2017.0.098/
root@localhost:~# ./install_GUI.s h
```

Figura A.1: *Pantalla de carga de Intel® MKL.*

2. Revisar el cuadro de diálogo **Requisitos previos**, distribución Linux de x64, el soporte de compiladores de C/C++ y FORTRAN para Linux, soporte de MKL para la distribución de Linux, implementación de MPI (MPICH2 versión 1.5 o Open MPI 1.8.x). Con esto se verifica si el sistema está configurado correctamente para la instalación, puede continuar con la instalación si están ausente los prerrequisitos opcionales. Sin embargo, si hace falta corregir un requisito crítico, debe antes corregirlo para así poder continuar. Cuando se termine haga clic en el botón **Siguiente**.

3. Elegir una de las opciones de activación como se muestra en la Figura A.2; seleccionando alguno de los botones de las opciones correspondientes:

   - **Utilizar licencia existente**: seleccione esta opción si tiene un archivo de licencia válido.
   - **Utilice el número de serie para activar e instalar el producto:** elija esta opción si tiene un número de serie. Esta opción requiere una conexión a Internet.
   - **Evalúe este producto** (no se requiere número de serie): elija esta opción si no lo hace Tiene un número de serie o si desea activar el producto más adelante. Esta opción no es Disponible para las versiones Beta.
   - **Elegir activación alternativa**: seleccione esta opción si desea realizar una de las siguiendo:
     - Activar offline (utilizando un archivo de licencia)
     - Usar un administrador de licencias

Cuando termine, hacer clic en **Siguiente**.

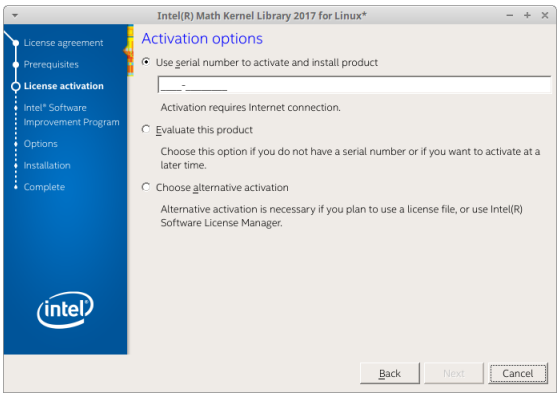


Figura A.2: *Pantalla de activación del producto.*

4. Elegir si desea participar en el Programa Intel® de mejora de software. Si opta por participar, Intel puede recibir automáticamente información anónima acerca de cómo usa sus futuros productos de desarrollo de software Intel®. Puede dejar de participar en cualquier momento desde Intel® Software Manager. En la Figura A.3 se muestra este paso.

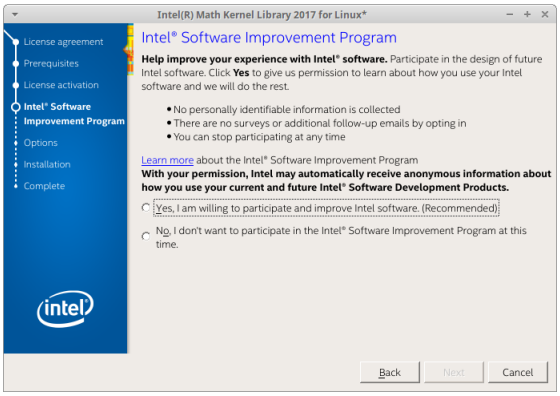


Figura A.3: *Programa de mejora de intel®.*

5. Revisar el **resumen de la instalación** para ver si las opciones de instalación seleccionadas cumplen con sus necesidades. Para cambiar la selección, haga clic en el botón **Personalizar la instalación** y proceder a cambiar las siguientes opciones:

   - Elegir un directorio de instalación alternativo, como se muestra en la Figura A.4.

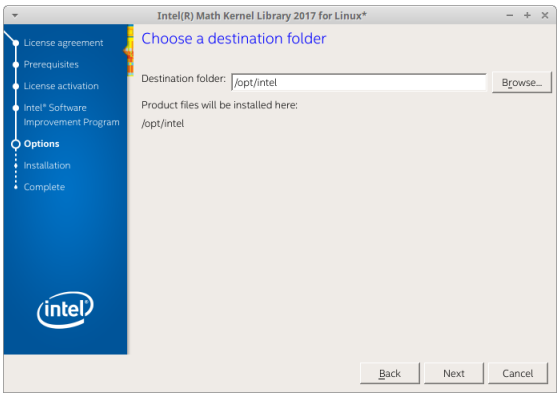


Figura A.4: *Carpeta de instalación.*

- Seleccionar la(s) arquitectura(s) para ejecutar sus aplicaciones. De forma predeterminada, IA-32 e Intel® 64 se han seleccionado como arquitectura.

  - Seleccionar los componentes que desea instalar, en la Figura A.5 se selección los componentes recomendados para compatibilidad con C/C++ como de FORTRAN.

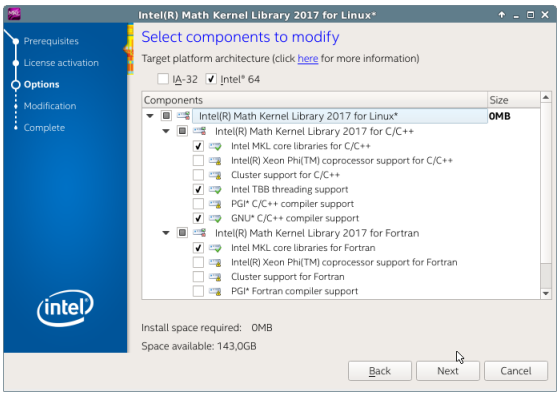


Figura A.5: *Componentes del paquete.*

Para continuar con la instalación, haga clic en el botón **Instalar**, en la Figura A.6 se presenta el recuento de las características de instalación.

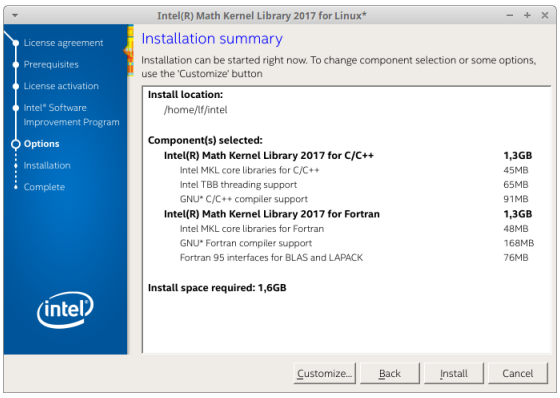


Figura A.6: *Recuento de las características seleccionadas para instalar.*

6. Hacer clic en **Siguiente** para iniciar la instalación. Cuando todos los componentes seleccionados estén instalados se muestra la ventana de finalización.

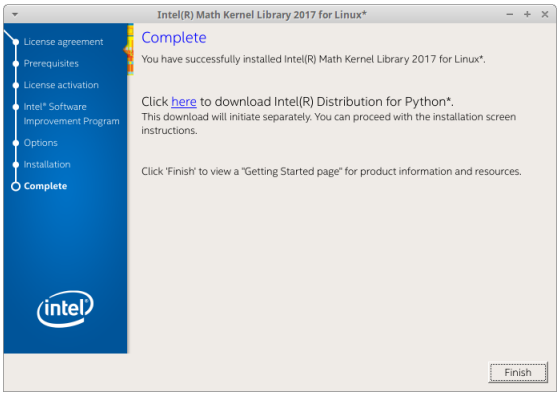


Figura A.7: *Finalizar la instalación.*

Además de lo especificado hasta el momento, se puede complementar la información con la lectura del manual de MKL[2] .

A.1.2. Eigen

Eigen es una biblioteca C++ de alto nivel de plantillas para álgebra lineal, operaciones de matrices y vectores, transformaciones geométricas, solucionadores numéricos y algoritmos relacionados. Eigen es una biblioteca de código abierto con licencia bajo MPL2 a partir de la versión 3.1.1. Las versiones anteriores fueron licenciadas bajo LGPL3+.

[2] https://software.intel.com/en-us/intel-mkl/documentation

Instalación Eigen

Método 1: Instalación sin utilizar CMake Se puede utilizar de inmediato los encabezados en el subdirectorio **Eigen**/. Para instalar, simplemente copiar el subdirectorio **Eigen**/ a su ubicación favorita. Si también desea las funciones no admitidas, copiar los archivos del directorio **unsupported**/.

Método 2. Instalación mediante CMake Llamar al directorio que contiene todos los archivos como '**source_dir**' (donde está el archivo INSTALL). Antes de comenzar, cree otro directorio al que se llamara '**build_dir**'.

Hacer:

```
user@localhost:~$ cd build_dir
user@localhost:~$ cmake ../../source_dir
root@localhost:~# make install
```

El paso "**make install**" puede requerir privilegios de administrador.

Puede ajustarse el destino de la instalación pasando la opción -DCMAKE_INSTALL_-PREFIX = myprefix a CMake, tal como está explicado en el mensaje que CMake imprime al final. Para mas información consultar en su dirección web[3] .

A.1.3. Lapack

LAPACK (**L**inear **A**lgebra **P**ackage) es una biblioteca de software estándar para álgebra lineal. Proporciona rutinas para resolver sistemas de ecuaciones lineales y mínimos cuadrados lineales, problemas de valores propios y descomposición de valores singulares. También incluye rutinas para implementar las factorizaciones de matrices asociadas tales como LU, QR, Cholesky y descomposición de Schur. LAPACK fue originalmente escrito en FORTRAN 77, pero se trasladó a Fortran 90 en la versión 3.2 (2008). Las rutinas manejan tanto matrices reales como complejas en precisión simple y doble.

[3] http://eigen.tuxfamily.org/index.php

Instalación Lapack

Se llamara al directorio que contiene todos los archivos '**source_dir**'. Antes de comenzar, crear otro directorio al que se llamara '**build_dir**'.

```
user@localhost:~$ cd build_dir
user@localhost:~$ cmake ../../source_dir
root@localhost:~# make install
```

Para mas información de Lapack consultar en la página web[4] .

A.1.4. OpenCV

OpenCV es una biblioteca libre de visión artificial originalmente desarrollada por Intel. Desde que apareció su primera versión alfa en el mes de enero de 1999, se ha utilizado en infinidad de aplicaciones. Desde sistemas de seguridad con detección de movimiento, hasta aplicaciones de control de procesos donde se requiere reconocimiento de objetos. Esto se debe a que su publicación se da bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas.

OpenCV es multi plataforma, existiendo versiones para GNU/Linux, Mac OS X y Windows. Contiene más de 500 funciones que abarcan una gran gama de áreas en el proceso de visión, como reconocimiento de objetos (reconocimiento facial), calibración de cámaras, visión estérea y visión robótica.

Instalación OpenCV

La descarga de OpenCV puede realizarse desde su página de internet[5] , como también desde los repositorios de Linux. Si la descarga se realiza desde la página con formato tar.gz, habrá que descomprimirla con:

```
user@localhost:~$ tar xvzf opencv-3.2.0.tar.gz
user@localhost:~$ cd opencv-3.2.0
```

[4] http://www.netlib.org/lapack/
[5] http://opencv.org/downloads.html

Después se consultara el fichero INSTALL, y seguir las instrucciones que ahí se explican. Según el documento, hay que tener en cuenta que se necesitan tener instaladas unas librerías, para la cual se recomienda hacer uso de **synaptic** (gestor de paquetes gráfico).

```
user@localhost:~$ ./configure
user@localhost:~$ make
root@localhost:~# make install
```

A.1.5. QT

Qt es un framework multi plataforma orientado a objetos ampliamente usado para desarrollar programas que utilicen interfaz gráfica de usuario, así como también diferentes tipos de herramientas para la línea de comandos y consolas para servidores que no necesitan una interfaz gráfica de usuario.

Instalación QT

Para quienes no puedan instalar Qt Creator desde repositorios existe el método oficial por el cual se descarga un archivo .run; desde la página oficial de Qt[6], el cual se encarga de instalar QT Creator.

```
root@localhost:~# chmod + x qt-creator-opensource-linux-x86_64-4.1.0.run
root@localhost:~# ./qt-creator-opensource-linux-x86_64-4.1.0.run
```

Tras esto comenzará la instalación de QT, comenzando primero por la descarga. Así que para este método se necesita tener conexión a Internet.

[6] https://www.qt.io/download/

A.2. Manual de Uso.

La aplicación presenta una interfaz muy fácil de utilizar, en las Figuras A.8 y A.9 se muestran las áreas de trabajo y las configuraciones que se permiten hacer en la interfaz.

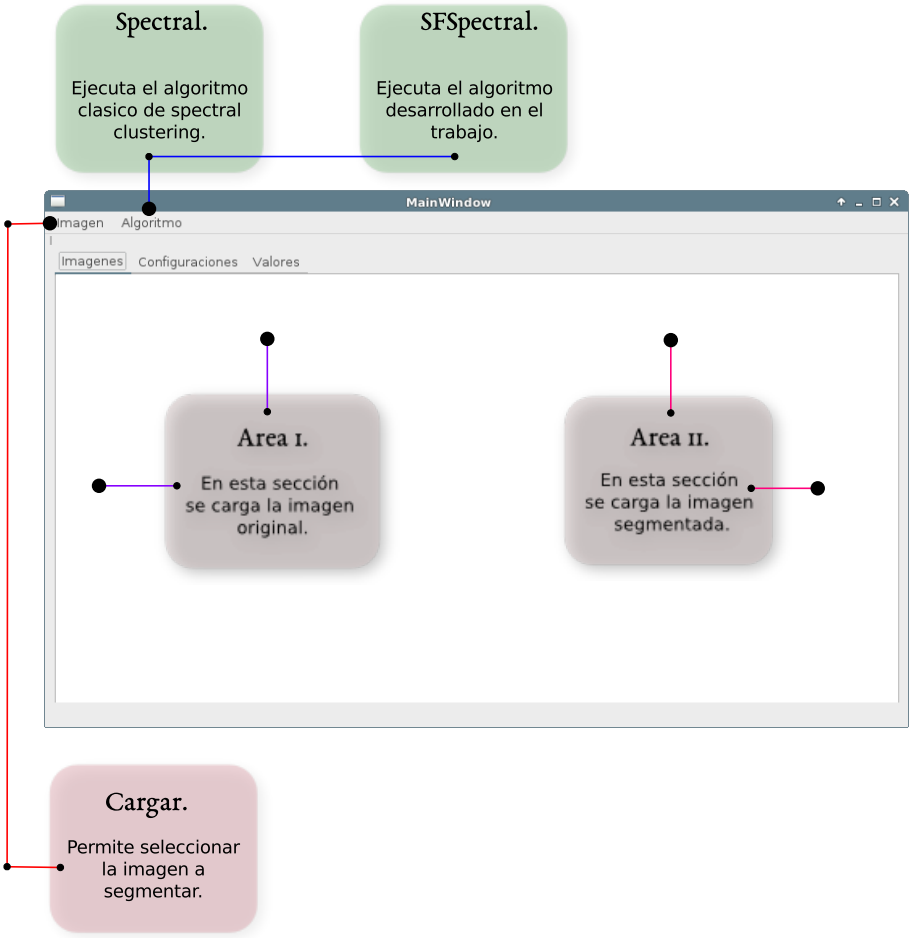


Figure A.8: *Áreas de trabajo de la aplicación.*

En la Figura A.8 se muestran las diferentes áreas de trabajo, área donde se carga la imagen original y la imagen segmentada, las acciones de cargar y guardar la imagen a segmentar y la aplicación del algoritmo **SC** como de **SFSC**.

Ahora se presenta la Figura A.9, en la cual se muestran las posibles configuraciones.

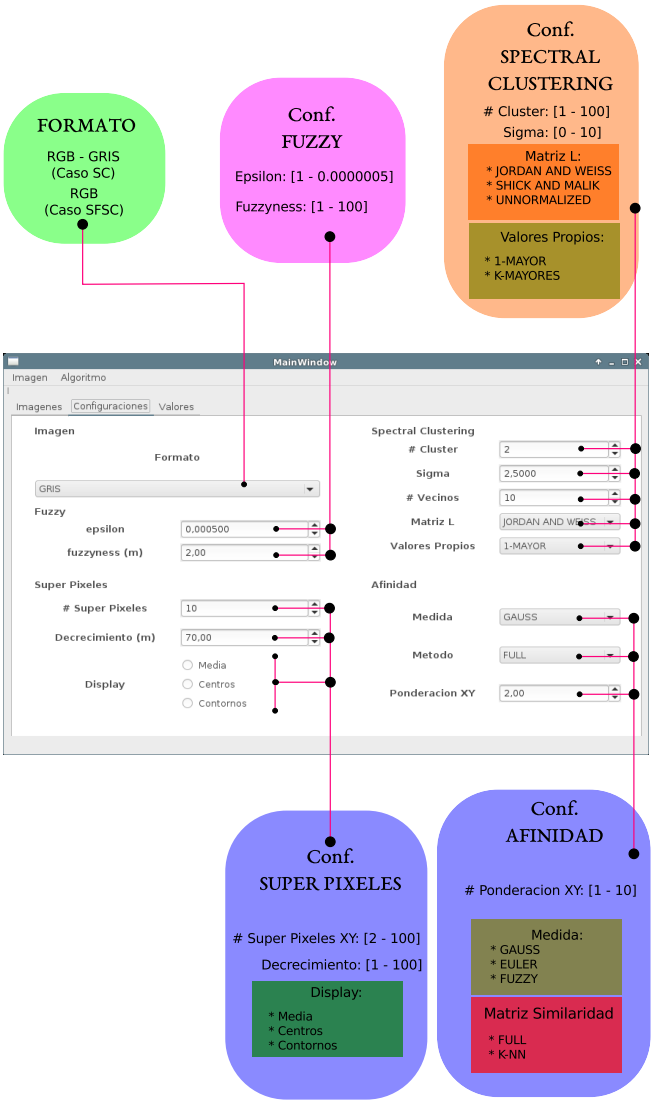
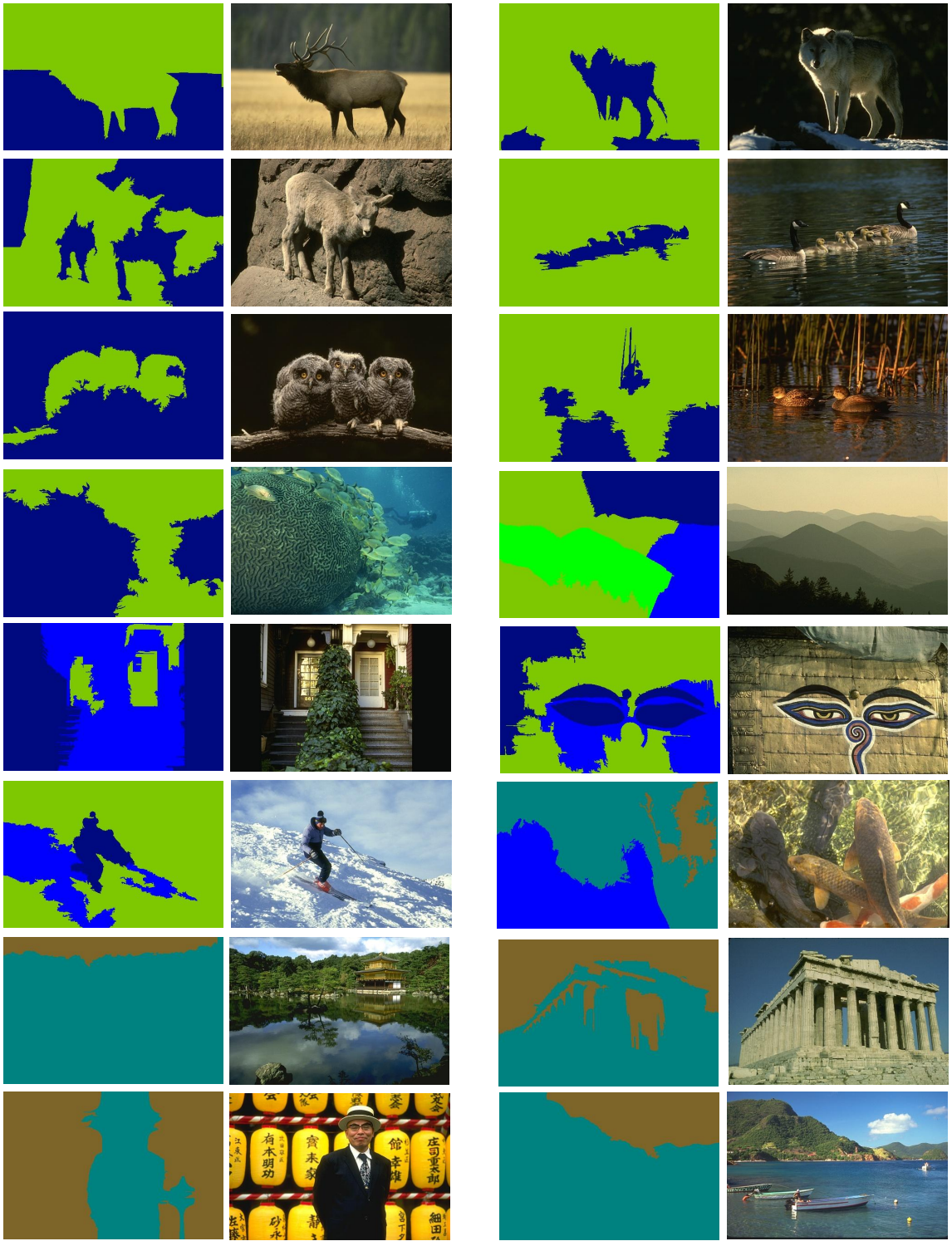


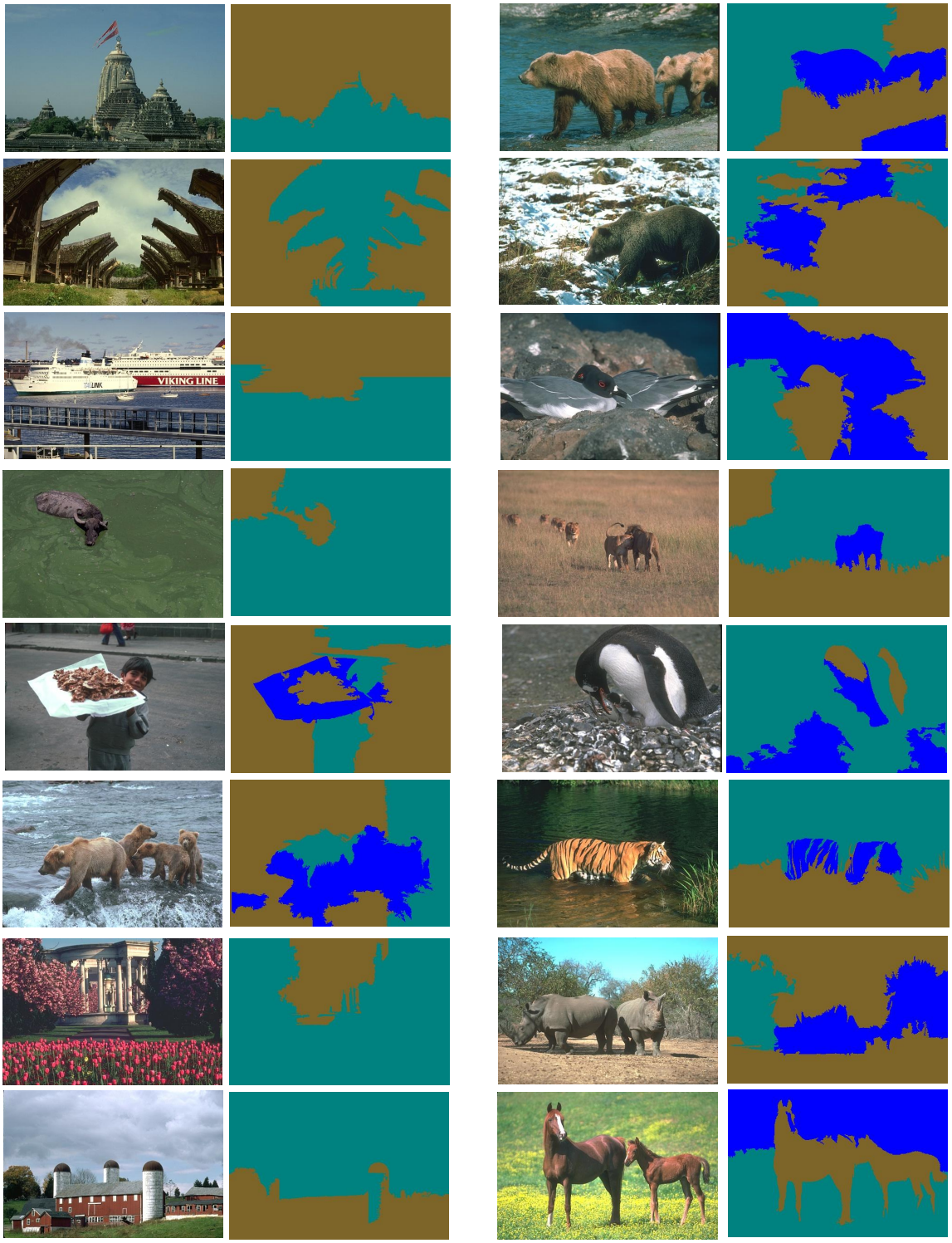
Figure A.9: *Parámetros de configuración.*

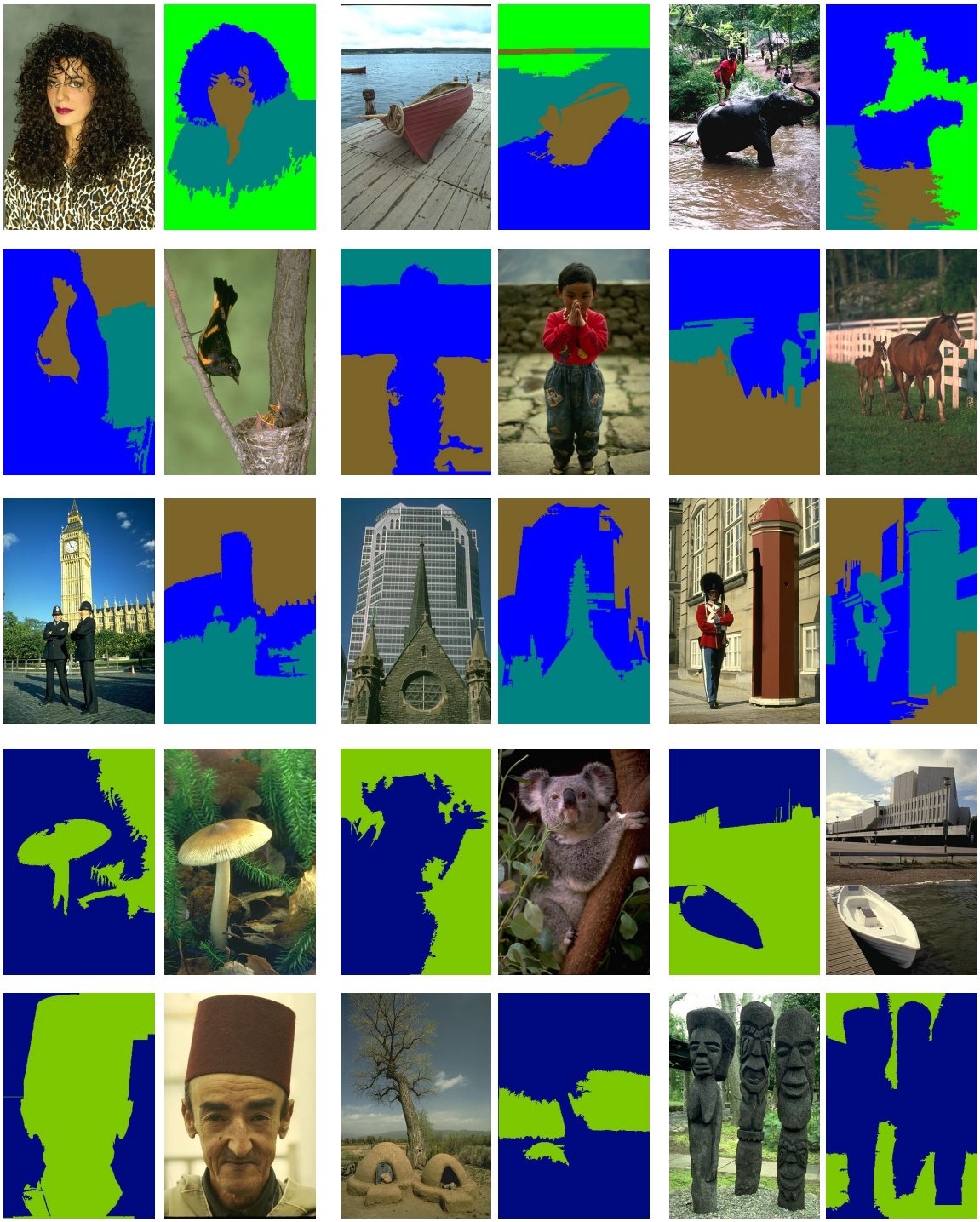
En la Figura A.9 se describen las diferentes posibilidades para los parámetros de configuración de los algoritmos **SC** y **SFSC**. Dentro de las configuraciones se encuentra la selección del tipo de imagen (Gris para **SC**, RGB para **SC** y **SFSC**), el tipo de matriz Laplaciana como de la matriz de similaridad, la función de distancia, cantidad de vectores propios y el tipo de visualización de la pre-segmentación por SLIC, además se pueden modificar los parámetros de las características anteriormente mencionadas, número de grupos como de súper píxeles, fuzzyness, epsilon, m, sigma, número de vecinos para la similaridad y factor de ponderación de la posición.

Capítulo B

Imágenes segmentadas




VIKING LINE



Bibliografía

[1] F. Zhao, L. Jiao, H. Liu, and X. Gao, "A novel fuzzy clustering algorithm with non local adaptive spatial constraint for image segmentation," *Signal Processing*, vol. 91, no. 4, pp. 988–999, 2011. [Online]. Available: http://dx.doi.org/10.1016/j.sigpro.2010.10.001

[2] S. Biswas, D. Ghoshal, and R. Hazra, "A new algorithm of image segmentation using curve fitting based higher order polynomial smoothing," *Optik - International Journal for Light and Electron Optics*, vol. 127, no. 20, pp. 8916–8925, 2016. [Online]. Available: http://dx.doi.org/10.1016/j.ijleo.2016.06.110

[3] M. Zhang, L. Jiao, W. Ma, J. Ma, and M. Gong, "Multi-objective evolutionary fuzzy clustering for image segmentation with MOEA/D," *Applied Soft Computing*, vol. 48, pp. 621–637, 2016. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1568494616303878

[4] X.-L. Jiang, Q. Wang, B. He, S.-J. Chen, and B.-L. Li, "Robust level set image segmentation algorithm using local correntropy-based fuzzy c-means clustering with spatial constraints," *Neurocomputing*, vol. 207, pp. 22–35, 2016. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0925231216302351

[5] Y. Yang, Y. Wang, and X. Xue, "A novel spectral clustering method with superpixels for image segmentation," *Optik*, vol. 127, no. 1, pp. 161–167, 2016. [Online]. Available: http://dx.doi.org/10.1016/j.ijleo.2015.10.053

[6] H. Liu, L. Jiao, and F. Zhao, "Non-local spatial spectral clustering for image segmentation," *Neurocomputing*, vol. 74, no. 1-3, pp. 461–471, 2010. [Online]. Available: http://dx.doi.org/10.1016/j.neucom.2010.08.021

[7] L. Wang and M. Dong, "Multi-level Low-rank Approximation-based Spectral Clustering for image segmentation," *Pattern Recognition Letters*, vol. 33, no. 16, pp. 2206–2215, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.patrec.2012.07.024

[8] C. G. Gavidia, "Segmentación De Imágenes Médicas Mediante Algoritmos De Colonia De Hormigas," Tesis de magister en Informatica, Pontificia Universidad Católica Del Perú, 2014. [Online]. Available: http://tesis.pucp.edu.pe/repositorio/handle/123456789/5619

[9] X. Li, J. Song, F. Zhang, X. Ouyang, and S. Khan, "MapReduce-based fast fuzzy c-means algorithm for large-scale underwater image segmentation," *Future Generation Computer Systems*, vol. 65, pp. 90–101, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X16300486

[10] X. Hong, J. Wang, and G. Qi, "Comparison of spectral clustering, K-clustering and hierarchical clustering on e-nose datasets: Application to the recognition of material freshness, adulteration levels and pretreatment approaches for tomato juices," *Chemometrics and Intelligent Laboratory Systems*, vol. 133, pp. 17–24, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.chemolab.2014.01.017

[11] D. Yang, R. Fei, J. Yao, and M. Gong, "Two-stage SAR image segmentation framework with an efficient union filter and multi-objective kernel clustering," *Applied Soft Computing*, vol. 44, pp. 30–44, 2016. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1568494616300473

[12] S. Zeng, X. Tong, and N. Sang, "Study on multi-center fuzzy C-means algorithm based on transitive closure and spectral clustering," *Applied Soft Computing Journal*, vol. 16, pp. 89–101, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.asoc.2013.11.020

[13] X. Baia, Z. Caoa, Y. Wanga, M. Yea, and L. Zhua, "Image segmentation using modified SLIC and Nyström based spectral clustering," *Optik - International Journal for Light and Electron Optics*, vol. 125, no. 16, pp. 4302–4307, 2014. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0030402614004471

[14] S. Zeng, R. Huang, Z. Kang, and N. Sang, "Image segmentation using spectral clustering of Gaussian mixture models," *Neurocomputing*, vol. 144, pp. 346–356, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.neucom.2014.04.037

[15] F. Zhao, H. Liu, and L. Jiao, "Spectral clustering with fuzzy similarity measure," *Digital Signal Processing: A Review Journal*, vol. 21, no. 6, pp. 701–709, 2011. [Online]. Available: http://dx.doi.org/10.1016/j.dsp.2011.07.002

[16] T. Inkaya, "A parameter-free similarity graph for spectral clustering," *Expert Systems with Applications*, vol. 42, no. 24, pp. 9489–9498, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2015.07.074

[17] L. R. Lorenti, "Segmentación espectral de imágenes obtenidas con cámaras de tiempo de vuelo," Tesis de Licenciatura en Informatica, Universidad Nacional de La Plata, 2014. [Online]. Available: http://sedici.unlp.edu.ar/handle/10915/34184

[18] J. Hou, W. Liu, and H. Cui, "Towards parameter-independent data clustering and image segmentation," *Pattern Recognition*, vol. 60, pp. 25–36, 2016. [Online]. Available: http://dx.doi.org/10.1016/j.patcog.2016.04.015

[19] H. Liu, F. Zhao, and L. Jiao, "Fuzzy spectral clustering with robust spatial information for image segmentation," *Applied Soft Computing Journal*, vol. 12, no. 11, pp. 3636–3647, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.asoc.2012.05.026

[20] Y. Yang and Y. Wang, "Simulated annealing spectral clustering algorithm for image segmentation," *Journal of Systems Engineering and Electronics*, vol. 25, no. 3, pp. 514–522, 2014. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6850231&isnumber=6850209

[21] H. Chang and D.-y. Yeung, "Robust path-based spectral clustering with application to image segmentation," *Tenth IEEE International Conference on Computer Vision (ICCV'05)*, vol. 1, pp. 278–285, 2005. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1541268

[22] Q. Li, Y. Ren, L. Li, and W. Liu, "Fuzzy based affinity learning for spectral clustering," *Pattern Recognition*, vol. 60, pp. 531–542, 2016. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0031320316301285

[23] Z. Li and J. Chen, "Superpixel Segmentation using Linear Spectral Clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274–2281, 2012. [Online]. Available: http://ieeexplore.ieee.org/document/7298741/

[24] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk, "SLIC Superpixels Compared to State-of-the-Art Superpixel Methods," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 11, pp. 2274–2282, 2011. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6205760papers3://publication/doi/10.1109/TPAMI.2012.120

[25] A. Schick, M. Fischer, and R. Stiefelhagen, "An evaluation of the compactness of superpixels," *Pattern Recognition Letters*, vol. 43, no. 1, pp. 71–80, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.patrec.2013.09.013

[26] X. Fan, L. Ju, X. Wang, and S. Wang, "A fuzzy edge-weighted centroidal Voronoi tessellation model for image segmentation," *Computers and Mathematics with Applications*, vol. 71, no. 11, pp. 2272–2284, 2016. [Online]. Available: http://dx.doi.org/10.1016/j.camwa.2015.11.003

[27] D. Xu, Z. Xu, S. Liu, and H. Zhao, "A spectral clustering algorithm based on intuitionistic fuzzy information," *Knowledge-Based Systems*, vol. 53, pp. 20–26, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.knosys.2013.07.020

[28] M. Filippone, F. Camastra, F. Masulli, and S. Rovetta, "A survey of kernel and spectral methods for clustering," *Pattern Recognition*, vol. 41, pp. 176–190, 2008. [Online]. Available: http://dl.acm.org/citation.cfm?id=1285173

[29] R. Diestel, *Graph Theory*, 5th ed. New york: springer-Verlag, 2000. [Online]. Available: http://diestel-graph-theory.com/index.html

[30] S. E. Schaeffer, "Graph clustering," *Computer Science review*, vol. 1, pp. 27–64, 2007. [Online]. Available: http://doi.org/10.1016/j.cosrev.2007.05.001

[31] A. Kaveh, "Introduction to Graph Theory and Algebraic Graph Theory," *Optimal Analysis of Structures by Concepts of Symmetry and Regularity*, pp. 15–35, 2013. [Online]. Available: http://dx.doi.org/10.1007/978-3-7091-1565-7_2

[32] F. M. Donald, "Models for spectral clustering and their applications," Ph.D. dissertation, Universidad de colorado, 2012. [Online]. Available: https://books.google.com.co/books?id=8aZEAQAACAAJ

[33] A. E. Brouwer and W. H. Haemers, *Spectra of graphs*. Springer, 2011. [Online]. Available: http://www.springer.com/us/book/9781461419389

[34] B. Mohar, "The laplacian spectrum of graphs," in *Graph Theory, Combinatorics, and Applications*, vol. 2. Ljubljan, Yuguslavia: Wiley, 1991, pp. 871–898. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.96.2577

[35] R. C. Fan, *Spectral Graph Theory*, Pennsylvania, Philadelphia, PA, 1997, vol. 92. [Online]. Available: http://www.math.ucsd.edu/~fan/research/revised.html

[36] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000. [Online]. Available: http://ieeexplore.ieee.org/document/868688/

[37] O. Cuadros Linares, "Segmentação de imagens de alta dimensão por meio de algoritmos de detecção de comunidades e super pixels," Ph.D. dissertation, ICMC - Universidade de São Paulo, 2013. [Online]. Available: http://www.teses.usp.br/teses/disponiveis/55/55134/tde-25062013-100901/

[38] X. Ren and J. Malik, "Learning a Classification Model for Segmentation," in *Computer Vision, 2003. Proceedings. 5th IEEE International Conference on*, vol. 1. Nice, Francia: IEEE, 2003. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1238308&isnumber=27772

[39] D. J. MacKay, "Information Theory, Inference, and Learning Algorithms," *Cambridge University Press*, pp. 284–292, 2003. [Online]. Available: http://www.inference.phy.cam.ac.uk/itila/p0.html

[40] commission on Illumination International, "CIE 15: Technical Report: Colorimetry," International commission on illumination, Washington, Tech. Rep., 2004. [Online]. Available: http://www.cdvplus.cz/file/3-publikace-cie15-2004/

[41] A. Radhakrishna, S. Appu, S. Kevin, L. Aurelien, F. Pascal, and S. Susstrunk, "SLIC Superpixels," *School of Computer and Communication Sciences*, 2010. [Online]. Available: http://www.kev-smith.com/papers/SLIC_Superpixels.pdf

[42] C. Cevahir and A. Aydin, "Efficient graph-based image segmentation via speeded-up turbo pixels," in *proceedings of 2010 IEEE 17th international conference on image processing*. Hong Kong: IEEE, 2010, pp. 3013–3016. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5653963&isnumber=5648792

[43] S. Das, "Pattern Recognition using the Fuzzy c-means Technique," *International journal of energy, information and communications*, vol. 4, no. 1, pp. 1–14, 2013. [Online]. Available: http://www.sersc.org/journals/IJEIC/vol4_Is1/1.pdf

[44] X. Zhang and Q. You, "An improved spectral clustering algorithm based on random walk," *Frontiers of Computer Science in China*, vol. 5, no. 3, pp. 268–278, 2011. [Online]. Available: http://dx.doi.org/10.1007/s11704-011-0023-0

[45] N. Andrew, M. Jordan, and Y. Weiss, in *Advances in Neural Information Processing Systems*. MIT Press, pp. 849–856. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.19.8100

[46] L. Hogben, "Spectral Graph Theory and the Inverse Eigenvalue Problem of a Graph," *Chamchuri journal of mathematics*, vol. 1, no. 1, pp. 51–72, 2009. [Online]. Available: http://www.match.sc.chula.ac.th/cjm

[47] J. Ponce and D. Forsyth, *Computer Vision a Modern Approach*, New york, 2012, vol. 2. [Online]. Available: https://www.pearsonhighered.com/program/Forsyth-Computer-Vision-A-Modern-Approach-2nd-Edition/PGM111082.html

[48] D. Arthur and S. Vassilvitskii, "How Slow is the k -Means Method? (Original title On the Worst Case Complexity of the k-means Method)." Stanford University, Arizona, Tech. Rep., 2006. [Online]. Available: http://theory.stanford.edu/~sergei/papers/kMeans-socg.pdf

[49] D. M. Kanungo Tapas, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Wu, "An Efficient K-Means Clustering Algorithm: Analysis and Implementation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 881–892, 2002. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1017616&isnumber=21893

[50] M.-s. Yang, P.-y. Hwang, and D.-h. Chen, "Fuzzy clustering algorithms for mixed feature variables," *Fuzzy sets and systems*, pp. 301–317, 2004. [Online]. Available: https://doi.org/10.1016/S0165-0114(03)00072-1

[51] F. kovacs, L. Babos, and A. Babos, "Cluster Validity Measurement Techniques," Budapest University of Technology and economics, Budapest, Hungria, Tech. Rep., 2016. [Online]. Available: https://pdfs.semanticscholar.org/c4f9/df3c66105382d05e58ec35faa8d435f55c91.pdf

[52] R. Kannan, S. Vempala, and A. Vetta, "On Clusterings: Good, Bad and Spectral," in *Foundations of Computer Science, 2000.*, vol. 51, no. 3. Redondo Beach, CA, USA, USA: IEEE, 2004, pp. 497–515. [Online]. Available: http://ieeexplore.ieee.org/document/892125/

[53] H. Zhang, J. E. Fritts, and S. A. Goldman, "Image segmentation evaluation: A survey of unsupervised methods," vol. 110, pp. 260–280, 2008. [Online]. Available: http://doi.org/10.1016/j.cviu.2007.08.003

[54] T. Fawcettt, "ROC Graphs: Notes and Practical Considerations for Researchers," *Kluwer academic publisher*, no. 1, pp. 1–38, 2004. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.10.9777

[55] E. Amigo, J. Gonzalo, J. Artiles, and F. Verdejo, "A comparison of Extrinsic Clustering Evaluation Metrics based on Formal Constraints," *Springer*, vol. 12, no. 4, p. 461, 2008. [Online]. Available: http://dx.doi.org/10.1007/s10791-008-9066-8

[56] R. M. William, "Objective Criteria for the Evaluation of Clustering Methods," *Journal of the American Statistical Association*, vol. 66, no. 336, pp. 846–850, 2009. [Online]. Available: htttp://www.jstor.org/stable/2284239

[57] J. M. Santos and M. Embrechts, *On the Use of the Adjusted Rand Index as a Metric for Evaluating Supervised Classification*, alippi, ce ed. Berlin, Heidelberg: springer, 2009, vol. 5769. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-04277-5_18

[58] S. Chaimontree, K. Atkinson, and F. Coenen, *Best Clustering Configuration Metrics: Towards Multiagent Based Clustering*. Liverpool, UK: Springer Berlin Heidelberg, 2010, vol. 6440, p. 12. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-17316-5_5/

[59] Q. Zhao and M. Xu, "Sum-of-Square Based Cluster Validity Index and Significance Analysis," in *Adaptive and Natural Computing Algorithms: 9th International Conference, ICANNGA 2009, Kuopio, Finland, April 23-25, 2009, Revised Selected Papers*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 313–322. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-04921-7_32

[60] D. L.Davies and D. W. Bouldin, "A Cluster Separation Mesure well-accepted similarity," *IEEE Transactions on pattern analysis and machine intelligence*, no. 2, pp. 224–227, 1979. [Online]. Available: www.ncbi.nlm.nih.gov/pubmed/21868852

[61] J. C. Dunn, "Well-Separated Clusters and Optimal Fuzzy Partitions," *Journal of Cybernetics*, no. June 2013, pp. 37–41, 2008. [Online]. Available: http://dx.doi.org/10.1080/01969727408546059

[62] M. Ujjwal, B. Sanghamitra, and M. Anirban, "Multiobjective Genetic Algorithms for Clustering - Applications in Data Mining and Bioinformatics," 2011. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-16615-0

[63] R. Guerequeta and A. Vallecillo, *Técnicas de Diseño de Algorito*, Málaga, Espain, 2000. [Online]. Available: http://www.lcc.uma.es/~av/Libro/

[64] R. Iakymchuk, "Performance Prediction through Time Measurements," Aachen, Alemania, Tech. Rep. 2, 2010. [Online]. Available: https://www.pdc.kth.se/~riakymch/pubs/hpc_ua.pdf

[65] F. Havet, "Graph Theory: Complexity of algorithms," in *Combinatorial Optimization - Algorithms for telecommunications*, 2007, ch. 3, pp. 29–39. [Online]. Available: http://www-sop.inria.fr/members/Frederic.Havet/

[66] C. Fowlkes, S. Belongie, and J. Malik, "Efficient Spatiotemporal Grouping Using the Nyström Method," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001.* Kauai, HI, USA, USA: IEEE, 2001. [Online]. Available: http://ieeexplore.ieee.org/document/990481/

[67] N. Seo, "Normalized Cuts and Image Segmentation," 2006. [Online]. Available: http://note.sonots.com/SciSoftware/NcutImageSegmentation.html

[68] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics," in *Proc. 8th Intel Conf. Computer Vision,*, 2001, pp. 416–423. [Online]. Available: https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/