5. Descripción del Algoritmo

1. Introducción

Desde la década pasada, la segmentación de imágenes en color ha suscitado una atención creciente. Las técnicas de segmentación de imágenes pueden clasificarse principalmente en los siguientes grupos:

- Técnicas basadas en la umbralización (histograma) y búsqueda de agrupaciones.
- Técnicas basadas en regiones.
- Técnicas basadas en la detección de bordes y discontinuidades.
- Técnicas basadas en la textura.

Un algoritmo de segmentación puede ser la combinación de varias de las técnicas anteriormente mencionadas.

Las técnicas basadas en histogramas suponen que las imágenes están compuestas por regiones con diferentes componentes de color. La ventaja de esta técnica de umbralización, es que no necesita información a priori de la imagen. Una de las desventajas es que no puede garantizar que las regiones sean contiguas, por no considerar los detalles espaciales. Este problema puede resolverse combinando esta técnica, con otra técnica de segmentación.

Las técnicas basadas en regiones, crecimiento de regiones, unión de regiones, división de regiones, y combinación de las dos últimas, se basan en formar regiones teniendo en cuenta la similitud entre los píxeles, y la posición espacial. Ésta es la principal ventaja de esta técnica. El principal inconveniente es la dependencia con la selección de los puntos *"semilla"* y con el orden en que se examinan las regiones de píxeles.

Las técnicas basadas en la detección de discontinuidades y bordes, presentan problemas cuando trabaja con imágenes que presentan bordes que están mal definidos, o que presentan demasiados bordes. Es posible obtener un buen resultado en la segmentación si combinamos técnicas basadas en regiones con técnicas basadas en la detección de bordes.

2. Descripción de Algoritmo

2.1 Introducción al algoritmo

En este proyecto se presenta un algoritmo de segmentación de imágenes en color que combina la técnica de crecimiento de regiones por adicción de píxeles, la clasificación según la información del color [7], y un proceso de etiquetado de relajación. Como ya se ha mencionado anteriormente, estas técnicas tienen sus propias ventajas y desventajas para distinguir regiones inconexas en imágenes en color. Algunas de las desventajas de una técnica pueden ser superadas por las ventajas de la otra técnica, por lo que la adecuada combinación de estos métodos produce mejores resultados en la segmentación.

Este algoritmo va a ser aplicado a imágenes médicas muy específicas: imágenes de quemados. El algoritmo debe funcionar correctamente en una amplia gama de imágenes diferentes, pero deben obtenerse resultados óptimos para esta aplicación Estas imágenes tienen unas características especiales:

- Son imágenes generalmente formadas por un fondo y un objeto central, que será el miembro a tratar. Sobre este elemento será donde habrá que centrarse para realizar la segmentación, intentando que el fondo quede segmentando en una única region.

- Generalmente, se quieren localizar zonas extensas, no pocos píxeles como sería el caso de la búsqueda de lunares para detectar posibles melanomas.

- El fondo de estas imágenes no va a ser homogéneo, ya que normalmente, el fondo estará formado por sábanas, servilletas, que harán que sean desigual.

- La presencia de sombras, y por tanto diferentes tonalidades, va a provocar segmentaciones no deseadas.

2.2. Objetivo del algoritmo

Como ya se ha visto, el objetivo principal de este algoritmo es distinguir las zonas afectadas por las quemaduras, de las zonas de piel sana, de forma que se aproxime a las segmentación que realiza el ojo humano.

Es decir, no se trata de conseguir mayor número de regiones segmentadas, sino de conseguir una segmentación coherente como la que realiza el ojo humano.

Se muestran a continuación dos imágenes, resultado de aplicar el algoritmo:

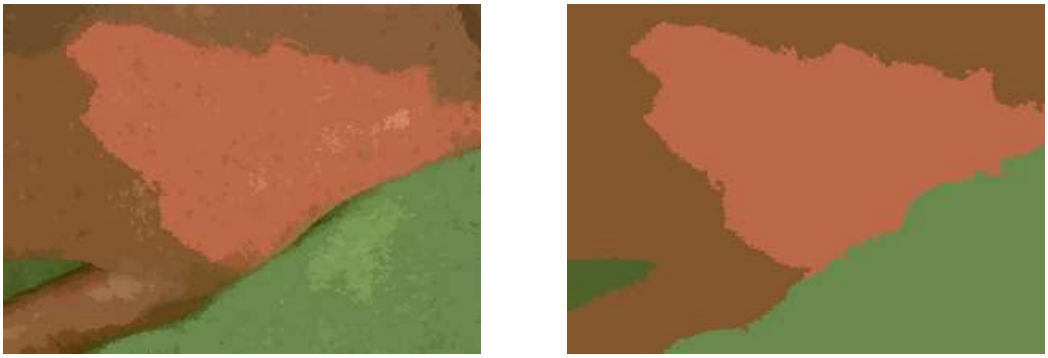

Figura 5.1: *Imágenes resultado de aplicar el algoritmo*

Como puede verse en la primera imagen, el número de regiones en la que está segmentada es mucho mayor que en la segunda. Es indudable, que los colores que la primera imagen discrimina son correctos, pero la información es redundante. El resultado de la segunda imagen, aunque no determina tantas zonas, es claro y completo.

2.3. Desarrollo del Algoritmo

A continuación se muestra un esquema general del algoritmo:

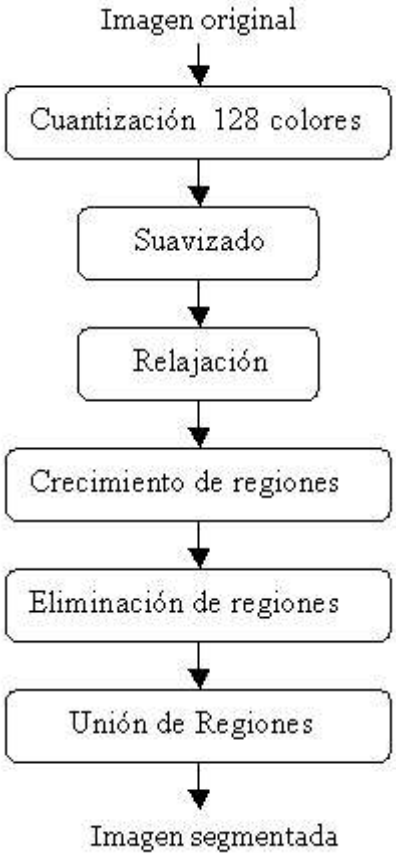

Figura 5.2: *Esquema general del algoritmo*

Se va a ir exponiendo cada uno de los pasos que se llevan a cabo en algoritmo.

2.3.1. Cuantización del color

El primer paso del algoritmo es cuantizar la imagen. Una buena cuantización es importante para el proceso de segmentación. En este trabajo, el modelo de representación utilizado es RGB. Generalmente, una imagen de color se representa con 24 bits, por lo que la imagen posee mil colores. Es por ello que hay que realizar la cuantización intentando no degradar las características del color.

Para ver cuál es la grado de cuantización óptimo, se compararan los resultados de cuantificar a 64, 128 y 256 colores. No se escogen valores superiores a 256, porque la función de Matlab utilizada, rgb2ing, no lo permite.

Si se observan las imágenes resultado del proceso de segmentación se puede concluir que:

- Si se cuantifica a 64 colores, se pierde bastante información de color, que para hacer la segmentación es necesaria.

- Si se cuantifica a 256 colores, cuando se realiza el proceso de segmentación, se obtienen regiones sobre-segmentadas.

Como consecuencia, en este proyecto se van a cuantizar las imágenes a 128 colores *(RGBorg)*.

chica75.jpg

Figura 5.3: *Imagen original*

Figura 5.4: *Imagen cuantizada a 128 colores*

2.3.2. Suavizado (Smooth)

El suavizado *(smooth)* es un procedimiento habitual en el pre-procesamiento de las imágenes; de hecho es una práctica habitual, cuando se va realizar una segmentación de crecimiento de regiones, porque se eliminan regiones que por sus características no añaden información, y en caso de no ser eliminadas producen sobre-segmentación.

La eliminación de estas zonas no debe afectar a la conservación de los contornos de los objetos que se quieren separar.

El proceso de suavizado o smooth consiste en eliminar, o al menos reducir el ruido de una imagen.

El proceso de Suavizado, consiste en:

1. Se toma una ventana W, formada por los NxN vecinos de p, siendo p un pixel de la imagen I que se ha seleccionado para el estudio del algoritmo.

2. Se clasifican los píxeles de la ventana W en dos clases, clase0 y clase1, usando el A*nálisis de las Componentes Principales* [4].

3. Se calculan las medias, $\overline{c_0}$ y $\overline{c_1}$, de la clase0 y clase1 respectivamente.

4. Sea $\overline{c'_p}$ la nueva etiqueta de color para el pixel p:

   - $\overline{c'_p} = \overline{c_0}$ si el número de píxeles pertenecientes a la clase0 es mayor que el número de píxeles que pertenece an la clase1.
   - $\overline{c'_p} = \overline{c_1}$ si el número de píxeles pertenecientes a la clase1 es mayor que el número de píxeles que pertenecen a la clase0.

5. Se devuelve el valor $\overline{c'_p}$.

El propósito de etiquetar p con el color representativo de una clase de mayor tamaño, en lugar de etiquetar p con el color representativo de la clase a la que él mismo pertenece, es eliminar el ruido.

2.3.2.1. Análisis de las Componentes Principales

El PCA se utiliza para obtener una representación simple y con menor dimensión para un conjunto de m variables correlacionadas. Las componentes principales se obtienen transformando las variables originales a un nuevo conjunto de variables no correlacionadas usando una rotación ortogonal en el espacio p-dimensional, las primeras componentes resumirán en orden decreciente la mayor cantidad posible de la variabilidad de las variables originales.

Como se ha indicado en el paso 2 del proceso de Suavizado, para clasificar los píxeles de la imagen en clase0 y clase1, se va a utilizar el análisis de las componentes principales (PCA). Este análisis va a permitir realizar una compresión de la imagen en color. El método propuesto para realizar dicho análisis se muestra a continuación:

Sea m el número total de píxeles pertenecientes a un bloque y $x_i' = (r_i, g_i, b_i)$ *siendo* $i = 1, \cdots, m$, *las componentes R,G y B de los distintos píxeles que componen la imagen.*

Calculamos el vector media y la matriz covarianza de los vectores de color x_i' *de la siguiente forma:*

$$\bar{x} = \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sum = \frac{1}{m}\sum_{i=1}^{m} x_i x_i' - \bar{x}\bar{x}'$$

El valor principal para el pixel i, vendrá dado por el vector y_i, *el cual se puede calcular a partir de la siguiente expresión:*

$$y_i = u'(x_i - \bar{x})$$

donde el vector u, se obtiene a partir de $\sum u = \lambda \cdot u$

Como la media de los valores principales, principal scores, es siempre 0, podemos usar esta característica para clasificar los píxeles en las dos clases:

- *Si* $y_i < 0$, *entonces* p_i *pertenece a la clase0.*
- *Si* $y_i > 0$, *entonces* p_i *pertenece a la clase1.*

La implementación de todas las funciones, y por tanto del algoritmo que en este proyecto se presenta se va a hacer utilizando Matlab, por lo que todo el proceso anterior para la clasificación de píxeles utilizando PCA, se puede realizar utilizando una función que tiene Matlab para este fin.

Por tanto, el proceso queda reducido a los siguientes pasos:

1. Se selecciona el pixel p, sobre el que se va a trabajar.
2. Se buscan los nxn vecinos del pixel p.
3. Se crea una matriz *matriz_pca,* que contendrá por columnas las componentes R,G y B de cada pixel vecino. Si se considera n=3 y p=(2,2), *matriz_pca* será de la siguiente forma:

$$matriz\_pca = \begin{bmatrix} R_{11} & G_{11} & B_{11} \\ R_{12} & G_{12} & B_{12} \\ R_{13} & G_{13} & B_{13} \\ R_{21} & G_{21} & B_{21} \\ R_{23} & G_{23} & B_{23} \\ R_{31} & G_{31} & B_{31} \\ R_{32} & G_{32} & B_{32} \\ R_{33} & G_{33} & B_{33} \end{bmatrix}$$

1. Una vez que se tiene la matriz, se utiliza la función de Matlab:

[COEFF, SCORE]=princomp(matriz_pca);

Esta función devuelve como resultado *score*, que es una matriz que contiene las principales componentes. Las columnas de *score* se corresponden con las observaciones, y las filas de *score* son las distintas componentes. Nos quedamos con la primera columna, y en función de los valores (si son mayores o menores que 0) se clasifican los píxeles en una de las dos clases (clase0 y clase1).

La función que se utilizará para obtener la imagen suavizada será *smooth.m*, y la matriz que se obtendrá como resultado se llamará *RGBlab*.

A continuación se muestra el resultado del proceso de suavizado:

Figura 5.5: *Imagen resultado del proceso de suavizado*

2.3.3. Proceso de Relajación

En primer lugar, se calcula la matriz de distancias *(calcula_distancias.m)*. Esta matriz es el resultado de calcular las distancias euclídeas, entre *RGBorg* (imagen original cuantizada a 128 colores), y *RGBlab* (imagen resultado del proceso de suavizado o smooth), recorriendo las matrices pixel a pixel.

Se define para cada pixel p, un parámetro:

$$d[p] = D(\vec{c}_p^{\,org}, \vec{c}_p^{\,lab})$$

donde D(*,*) representa la distancia euclídea entre dos vectores. Por lo tanto, en nuestro caso, d[p] representa la distancia euclídea entre el vector de color $\vec{c}_p^{\,org}$ de la imagen original, y el vector de color etiquetado, $\vec{c}_p^{\,lab}$, tras el proceso de Suavizado. Es decir:

$$d[p] = D(\vec{c}_p^{\,org}, \vec{c}_p^{\,lab}) = \sqrt{(R_{org} - R_{lab})^2 + (G_{org} - G_{lab})^2 + (B_{org} - B_{lab})^2}$$

Como resultado se obtiene la matriz *distancia_euclidea:*

$$\textbf{distancia\_euclidea}=\begin{bmatrix} d_{11} & \cdots & \cdots & d_{1n} \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ d_{m1} & \cdots & \cdots & d_{mn} \end{bmatrix}$$

Figura 5.6: *Representación de la distancia euclídea entre RGBorg y RGBlab.*

Se puede observar que las zonas más oscuras corresponden a las zonas en las que *RGBorg* y *RGBlab* son prácticamente iguales. Las zonas más claras, son consecuencia del proceso de suavizado, y se puede observar que "dibujan" el contorno del elemento.

Una vez que se tiene la matriz *distancia_euclidea*, y la matriz *RGBlab (*resultado del proceso de Smooth), se puede comenzar a mapear la imagen.

Para obtener *RGBlabmod* (imagen resultado del proceso de mapeo), se va recorriendo la matriz *RGBlab* pixel a pixel, y se comprueba si es posible realizar el proceso de relajación.

El proceso de relajar una pareja de vecinos (p,q), consiste en comprobar si se puede reducir el valor de *d[q]*, si ahora se utiliza p.

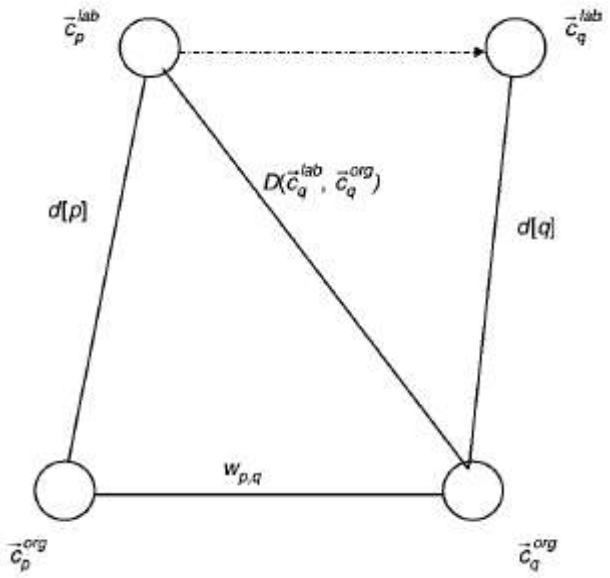

Figura 5.7: *Esquema general del proceso de relajación*

El proceso de relajar el par (p,q) se basa en comprobar si se verifica la relación:

$$d[q] > d[p] + w_{p,q}$$

donde q es un vecino de p y $w_{p,q}$ viene definido por $w_{p,q} = D(\vec{c}_p^{org}, \vec{c}_q^{org})$ y en ese caso, realizar los siguientes cambios:

- d[q] = d[p]
- $\vec{c}_q^{lab} = \vec{c}_p^{lab}$

Como resultado de este proceso se obtiene la matriz *RGBlabmod,* sobre la cual se realizará el proceso de segmentación.

Figura 5.8: *Representación de la imagen resultado del proceso de relajación*

2.3.4. Crecimiento de regiones

Para conseguir el objetivo de esta fase, la imagen segmentada, son necesarias dos matrices:

- La primera matriz es NxM (2 dimensiones), donde a cada pixel tiene asignado un número de región, de forma que lo píxeles que tengan el mismo numero de región, formarán una misma región.

- La segunda matriz es una tabla donde cada numero de región tiene asignado un color. El color que cada región tiene asignado es la media del color de todos los píxeles que forman una misma región.

Para obtener la imagen segmentada, se va a ir recorriendo la imagen pixel a pixel. El proceso a seguir es el siguiente:

Se toma el primer pixel (izquierda superior), p, y se le asigna el número de región 1. Se calculan las distancias euclídeas entre el pixel p y todos sus vecinos (q_1, q_2, q_3):

$$D_i(\vec{c}_p, \vec{c}_{qi}) = \sqrt{(R_p - R_{qi})^2 + (G_p - G_{qi})^2 + (B_p - B_{qi})^2} \leq umbral$$

Si se cumple la relación anterior, "q_i" pasará a formar parte de la region 1, en caso contrario, formará una nueva región.

A continuación, se toma un pixel que se haya unido a la región 1, y se realiza el mismo proceso con sus vecinos. Así, hasta que ya se haya hecho con todos los píxeles que forman parte de la región 1.

Una vez finalizado, se toma el siguiente pixel (izquierda-derecha, arriba-abajo), que aún no se haya analizado, hasta llegar al final de la imagen. Una vez que a un pixel se le asigne un número de región, ya no se podrá modificar.

Cuando finalizamos el proceso ya se tienen las dos matrices a las que anteriormente se hacía referencia.

Como se puede observar, esta fase depende de un valor que debe elegir el usuario: *Umbral*. Se puede decir, que esta es la fase menos automática del algoritmo. La forma de elegir este umbral es mediante prueba-error. Es decir, se establece un umbral, se aplica el algoritmo, y se analizan los resultados. Así, hasta obtener resultados satisfactorios.

A pesar de encontrar el umbral que produzca resultados óptimos, el proceso no termina aquí. En el 99% de los casos, el número de regiones en las que la imagen ha quedado segmentada es muy elevado; es decir hay regiones sobre-segmentadas. Esto es debido a que el algoritmo es muy sensible al umbral elegido. Si se elige un umbral muy elevado, no se segmenta bien la imagen, ya que prácticamente todos los píxeles quedan albergados en una misma región, y si se elige un umbral muy pequeño (muy restrictivo), se obtienen regiones sobre-segmentadas; es por esto, que hay que conseguir una solución de compromiso para obtener así resultados óptimos.

Figura 5.9: *Imagen resultado del proceso de crecimiento de regiones*

2.3.5. Eliminación de regiones

A continuación, se eliminan algunas regiones (se unen a aquellas regiones más cercanas en color) cuando el número de píxeles que contienen es inferior a la mitad de las dimensiones de la matriz a tratar (elimina_region.m). Esto es posible, ya que en general, se va a querer detectar regiones extensas, por tanto no serán de gran importancia las regiones pequeñas. Habrá que establecer un *minimo* de píxeles, para todo el algoritmo.

Se considerará como mínimo, la mitad de las filas o columnas de la imagen, es decir:

Minimo=min(Nf/2,Nc/2)

Una vez realizado este paso, el número de regiones en el que queda dividida la región se reduce considerablemente.

Con este paso hay que tener cuidado, ya que aunque como hemos dicho anteriormente, normalmente se van a querer detectar zonas extensas, hay ocasiones en las que si tenemos quemaduras muy pequeñas, y en este paso pueden desaparecer.

Figura 5.10: *Imagen resultado de eliminar las regiones pequeñas*

5.3.6. Unión de regiones

En esta fase se realiza la unión de regiones que están muy próximas en términos de color. Es decir, de nuevo se establece un umbral (prueba-error), y aquellas regiones cuya distancia euclídea sea menor que este cierto umbral, quedarán unidas en una única región (aquella formada por un número mayor de píxeles).

Figura 5.11: *Imagen resultado de unir las regiones próximas en color*

5.3.7. Selección de la zona deseada

Por último, se añade la posibilidad de seleccionar con el ratón, la zona en la cual se está interesado, y así obtener la segmentación de esa región. Para seleccionar la zona deseada, hay que marcar dos píxeles en la imagen original.

Figura 5.12: *Imagen resultado del algoritmo*