Introducción al tratamiento Digital y Clustering de imágenes

Por Juan Manuel Miguel Jiménez

Juan Manuel Miguel Jiménez
Departamento de Electrónica. Universidad de Alcalá.
jmanuel@depeca.uah

En este artículo se tratan de una manera general los algoritmos más típicos del procesado de imágenes, como la convolución, ecualización del histograma, umbralizado, negativo, etc. Además, se estudian los aspectos teóricos y la aplicación del algoritmo c-means al agrupamiento de imágenes, todo ello con un enfoque práctico, mientras se explica el desarrollo de una aplicación en Visual Basic en la que implementan todos estos métodos. Varias figuras ilustran el manejo de la aplicación y muestran algunos de los resultados obtenidos.

El trabajo que se presenta resume los conceptos más básicos relacionados con el tratamiento digital de imágenes, y describe el desarrollo de un programa para la implementación de los mismos. Además de abordar las operaciones más típicas propias del preprocesado de imágenes, se ha incluido la descripción de las técnicas de agrupamiento de objetos y la implementación del algoritmo c-means, aspecto al que este artículo dedica más énfasis. Para la realización de los algoritmos se ha elegido como lenguaje de programación Visual Basic, sencillo de programar y que ofrece una interface de usuario muy agradable al trabajar bajo Windows, así como objetos y métodos muy apropiados para visualizar gráficos.

Básicamente, el programa que se ha realizado integra las siguientes funciones:

- Lectura de ficheros en formato BMP, tanto en B/N como en color, y representación de los mismos.
- Representación del histograma de la imagen en B/N.
- Mejora del contraste mediante la ecualización del histograma tanto en B/N como en color.
- Umbralizado de imágenes en B/N con un valor arbitrario, para conseguir una imagen binaria.
- Obtención del negativo de una imagen en B/N.
- Mejora del contraste en zonas oscuras mediante la aplicación del logaritmo en imágenes en B/N.
- Cálculo del gradiente, utilizando las máscaras de Sobel.
- Detección de líneas, tanto horizontales como verticales, sobre imágenes en B/N.
- Modificación del brillo de las imágenes en B/N.
- Convolución de imágenes en B/N, para poder aplicar filtrados genéricos, introduciendo los valores de los coeficientes de la máscara desde el propio interface gráfico.
- Seleccionar colores en una imagen y umbralizarla en base a ellos, con objeto de aplicar un posterior proceso de agrupamiento basado en el algoritmo c-means.
- Técnica de agrupamiento, basada en c-means, con posibilidad de situar los Prototipos de forma gráfica(con el ratón) o numérica(con el teclado).
- Posibilidad de imprimir los resultados, o bien almacenarlos en formato BMP, tanto en B/N como en color.
- Deshacer el último cambio que se acaba de efectuar.

Operaciones Básicas de Procesado

Pasamos a describir brevemente algunas de las operaciones más elementales que integra nuestra aplicación software, y que se basan siempre en la aplicación directa de una sencilla función sobre el nivel de gris de los pixels de entrada. Representan las operaciones más típicas y simples en el procesado de imágenes.

Umbralizado de la Imagen

Una forma de separar los objetos del fondo en una imagen, o dicho de otra forma, segmentar los objetos del fondo, es umbralizar o binarizar la imagen. Básicamente, con este método elegimos un umbral y a todos los pixels cuyo nivel de gris se encuentre por encima de este valor se les asignará un nivel máximo de blanco, y al resto se les asignará un valor cero o negro. Para este fin tenemos el botón llamado Umbralizado, y es fácil imaginar el algoritmo que lo realiza: básicamente, un doble bucle para recorrer todas las filas y columnas de la matriz que contiene la imagen, y en el que se hace una comparación del nivel de gris del pixel actual con un umbral determinado.

Negativo de la Imagen

El efecto de negativo se consigue, básicamente, sustituyendo el nivel de gris de cada pixel por el de su complementario, esto es, L-r(x,y), donde L es el nivel máximo de gris (255 en nuestro caso) y r(x,y) es el nivel de gris del pixel actual que se está procesando. El procedimiento conducido por el evento click sobre el botón llamado Negativo es el encargado de realizar este algoritmo.

Logaritmo de una Imagen

También se ha dispuesto un botón para el cálculo del logaritmo. La aplicación de un logaritmo a los niveles de gris tiene como resultado un realce del contraste en las zonas oscuras, a espensas de un empobrecimiento del mismo en las zonas con mayor nivel de gris. La fórmula general para la aplicación de un logaritmo, en base 10 como es en nuestro caso, es la siguiente:

$$g(x,y)=K\cdot\log10(1+f(x,y))$$

donde K es la constante que fija el máximo nivel de salida, y la constante 1 asegura que los valores de la salida sean positivos.

Modificación del Brillo

Esta manipulación es la más sencilla, consistiendo únicamente en seleccionar un valor, con un botón de desplazamiento en nuestra aplicación, para sumárselo al nivel de gris de cada pixel de la imagen, consiguiendo un oscurecimiento o aclarado de la misma. La figura 1 muestra un aspecto general del interface gráfico dedicado a las operaciones típicas de preprocesado de imágenes.

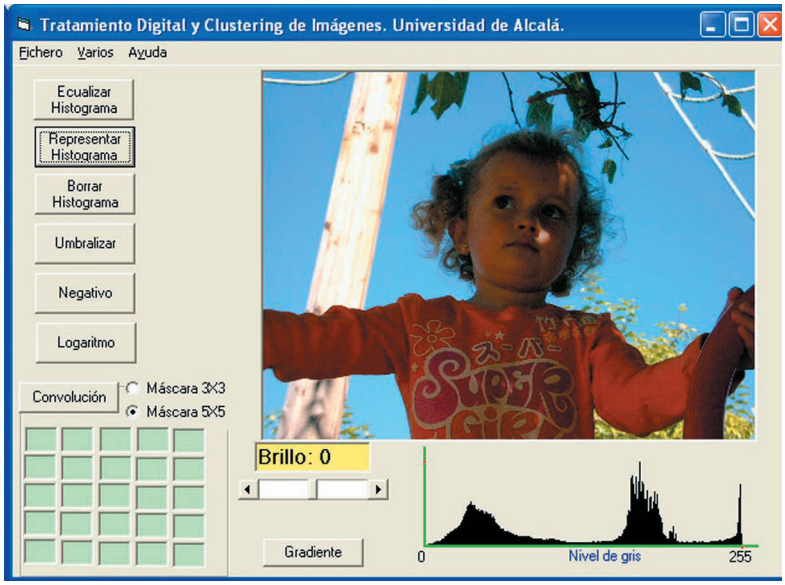

Figura 1. Aspecto general del interface gráfico para procesado de imágenes.

Histograma de una imagen

Mediante el histograma de una imagen conseguimos hacernos una idea aproximada de su distribución de niveles de gris y, por tanto, de su contraste. El histograma es una curva que en abcisas representa cada uno de los niveles de gris, mientras que en ordenadas representa la frecuencia relativa de aparición de ese nivel de gris. Así, una imagen cuyo histograma muestre que sus niveles de gris más abundantes se concentren en torno a un pequeño rango de valores, probablemente será una imagen poco contrastada, mientras que un histograma cuyos pixels presenten niveles de gris más o menos distribuidos de forma uniforme en el rango dinámico permitido, representará a una imagen fuertemente contrastada.

Calcular el histograma de una imagen, es simplemente contar el número de apariciones de cada uno de los posibles niveles de gris presentes en la misma. Si la imagen es en color, es posible extraer tres histogramas, cada uno de ellos correspondiente a un color primario. En nuestro caso, decidimos obtener previamente una imagen de luminancia sobre la que calculamos su histograma, paso previo que también se requiere para una posible y posterior ecualización de la imagen con idea de mejorar su contraste. En este último caso, también se obtienen dos matrices diferencia de color. Una vez ecualizada la matriz de luminancia, pasamos del formato YIQ anterior al RGB original. En nuestra aplicación, existe un procedimiento que se encarga de calcular y dibujar el histograma de las imágenes y se ejecuta cada vez que hacemos click en el botón correspondiente.

Ecualización del Histograma

Si una imagen presenta un histograma centrado en torno a un valor determinado, es probable que esté poco contrastada, aunque esto no es necesariamente cierto, ya que pueden existir imágenes que, por su naturaleza, presenten histogramas de este tipo. Existen diversos métodos estadísticos que logran mejorar el contraste, entre ellos, uno de los más conocidos se denomina ecualización uniforme del histograma y consiste en aplicar una función a la luminancia de los pixels, que consigue repartir la misma a lo largo de todo el rango dinámico de niveles de gris, y de una manera más uniforme.

Si asumimos que los valores de luminancia de los pixels son una variable aleatoria y que su valor puede fluctuar entre 0 y L, es decir, $0 \le r \le L$ y además asumimos que su función densidad de probabilidad es $p_r(r)$, conseguiremos que la imagen de salida tenga una densidad de probabilidad uniforme, si al nivel de gris de la imagen de entrada le aplicamos como función de transformación la función de distribución, $F_r(r)$, de ella misma.

$$F_r(r) = \int_0^r p_r(w)dw \quad ; 0 \le r \le L$$

donde w es una variable de integración ficticia.

Si aplicamos una formulación al caso de variables discretas, a partir del histograma que da el número de pixels con nivel de gris r_i, esto es, $h(r_i)$, podemos calcular la probabilidad de que un pixel tome un valor r_i de este modo:

$$p_r(r_i) = h(r_i)/n$$

donde n es el número de pixels de la imagen.

Con lo que, si se desea que los valores de salida, s_k, también presenten L niveles de gris, la transformación queda de la forma:

$$s_k = \sum_{i=0}^{k} p_r(r_i)$$

En nuestra aplicación existe un procedimiento que hace justamente todo esto para lograr ecualizar el histograma de la imagen, y se ejecuta cada vez que hacemos click en el botón correspondiente.

Operación de Convolución

Los procedimientos anteriores, así como los que se describirán más tarde para modificar el brillo, umbralizar, efecto negativo, etc, están basados en la aplicación de una función, más o menos sencilla, sobre el valor de la luminancia de cada uno de los pixels de la imagen a manipular. La convolución, en cambio, modifica la apariencia de cada pixel en función del valor de luminancia que toman los pixels de su entorno.

Según la teoría de sistemas, sabemos que estos pueden caracterizarse por su respuesta al impulso unitario, llamado $\delta(m)$ en el caso unidimensional y $\delta(m,n)$ en el bidimensional. En este último caso, como ocurre en las imágenes, la respuesta a un delta es la salida que presenta el sistema cuando a la entrada se aplica un punto de luminancia infinita y anchura nula. Así, la operación de convolución se define matemáticamente, para el caso bidimensional continuo como:

$$g(x,y) = h(x,y) \otimes f(x,y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h(x-x', y-y') \cdot f(x', y')dx'dy'$$

mientras que, en el caso discreto, las integrales se transforman en sumatorios de la manera siguiente:

$$g(m,n) = h(m,n) \otimes f(m,n) = \sum_{m'=-\infty}^{\infty} \sum_{n'=-\infty}^{\infty} h(m-m', n-n') \cdot f(m', n')$$

La convolución se realiza entre dos matrices: la matriz imagen y la máscara de convolución, obteniéndose un filtrado de acuerdo con esta última. Por motivos de carácter práctico, en nuestra aplicación se ha decidido ofrecer dos posibles tamaños de máscara: de 3x3 y de 5x5 coeficientes. Para obtener la convolución, se realiza el producto ponderado de la matriz de convolución con el entorno de un pixel, para cada pixel de la imagen(nosotros decidimos exceptuar los de los bordes de la imagen).

Cuando convolucionamos una imagen con una máscara estamos aplicando un filtro, cuyo tipo viene determinado por el contenido de la máscara. No es muy intuitivo imaginar qué tipo de filtrado se va a obtener partiendo de una determinada máscara, salvo en algunos casos especiales, como el filtrado de media, la detección de puntos y la de líneas horizontales y verticales, para cuyos casos los tipos de máscaras a aplicar son los siguientes:

Figura 3. Máscaras para los operadores de Sobel.

1	1	1
1	1	1
1	1	1

-1	-1	-1
-1	8	-1
-1	-1	-1

-1	-1	-1
2	2	2
-1	-1	-1

-1	2	-1
-1	2	-1
-1	2	-1

Figura 2. De izda. a dcha.: máscaras para filtrado de media, detección de puntos, líneas horizontales y verticales.

Gradiente de una Imagen

El gradiente es el método más empleado para calcular la derivada en una imagen. Para una función f(x,y), el gradiente de la misma en un punto de coordenadas (x,y) se define como el vector:

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

En la práctica, existen varias aproximaciones para el cálculo del gradiente de imágenes, todas ellas basadas en el módulo del vector gradiente:

$$|\nabla f| = \left[\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2 \right]^{1/2}$$

El método que empleamos en nuestra aplicación software consistió en la utilización de las llamadas máscaras de Sobel, que tienen la ventaja de proporcionar tanto una derivación como un efecto de suavizado, con lo que se consigue atenuar el ruido. Las derivadas parciales basadas en las máscaras de Sobel son:

$$G_x = (Z_7+2Z_8+Z_9) - (Z_1+2Z_2+Z_3)$$
$$G_y = (Z_3+2Z_6+Z_9) - (Z_1+2Z_4+Z_7)$$

luego se suman ambos valores para obtener una aproximación del gradiente. Las Z son los niveles de gris de los pixels solapados por las máscaras en cualquier posición de la imagen. Las máscaras que se utilizan para el cálculo de los operadores anteriores (operadores de Sobel), son las siguientes:

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

Z_1	Z_2	Z_3
Z_4	Z_5	Z_6
Z_7	Z_8	Z_9

Región imagen 3x3

De este modo, el cálculo del gradiente se reduce a aplicar la convolución con cada una de las máscaras anteriores, sumando luego sus resultados, operación que en nuestro caso lleva a cabo un procedimiento, al pulsar el correspondiente botón.

Agrupamiento mediante el Algoritmo C-means

Los algoritmos basados en el agrupamiento particional tienen como objetivo minimizar la varianza intracluster y maximizar la varianza intercluster. Estos métodos descomponen el conjunto de objetos en un conjunto de clusters disjuntos, minimizando una función criterio que enfatiza la estructura local de los objetos, asignando clusters a máximos locales en la estructura global. Son métodos que se usan típicamente en reconocimiento de formas.

Los algoritmos basados en el agrupamiento jerárquico proporcionan como resultado una secuencia anidada de grupos de pixels, que se puede representar en forma de árbol. Estos métodos se pueden llevar a cabo de dos formas distintas, uniendo pequeños clusters para formar otros mayores, o bien dividiendo clusters grandes en otros más pequeños.

Una vez elegida la medida de disimilitud que se vaya a emplear, se pasa a elegir el algoritmo de clustering. Los resultados que se obtienen con cada algoritmo pueden ser distintos, a pesar de que la imagen de partida y la medida de disimilitud empleadas sean las mismas. Además, diferentes valores de k (número de clusters) suelen generar diferentes tipos de clusters. La inicialización de los valores de los centroides de los clusters también es un hecho crítico, ya que algunos clusters podrían quedar vacíos si el centroide inicial queda lejos de la distribución de los objetos.

El algoritmo c-means es uno de los criterios de agrupamiento más ampliamente utilizado y se basa en la minimización de un índice de prestaciones, el cual se define como la suma de las distancias al cuadrado de todos los puntos u objetos incluidos en un cluster al centroide de dicho cluster. El número de clusters k es definido a priori. El comportamiento de este algoritmo depende del número de clusters especificado, la elección inicial de los centroides y el número de objetos en la imagen. Las consideraciones en las que se basa este algoritmo son:

- Las agrupaciones se pueden representar en todo instante por un solo punto, llamado prototipo de la clase, que generalmente se corresponde con el centro de masas de la agrupación.
- El error cometido al asignar un patrón a una clase depende de la distancia del patrón al prototipo de la clase.
- El objetivo es obtener una partición de los datos para la que se minimice el error cometido al clasificar todos los patrones del conjunto.

La función objetivo es:

$$J = \sum_{i=1}^{C} \sum_{x \in X_i} |x - m_i|^2$$

donde X_i es el conjunto de patrones que pertenece a la agrupación i-ésima y mi es el prototipo de esa agrupación. Las particiones obtenidas con este criterio reciben el nombre de particiones de mínima varianza.

Antes de aplicar una técnica de agrupamiento sobre una imagen en color, hemos decidido hacer una selección previa de los colores que nos interesan, con objeto de binarizar la imagen y que sólo queden los objetos de interés, es decir, aquellos objetos cuyos colores sean los seleccionados en una fase previa. Para elegir los colores en nuestra aplicación basta hacer click en el botón Seleccionar colores, evento que conduce a la ejecución de un procedimiento, el cual carga un nuevo formulario que nos interroga sobre el número de colores a seleccionar, y a continuación nos pide que los introduzcamos, bien rellenando numéricamente con el teclado cada una de las cajas de

texto correspondientes a las componentes RGB, bien gráficamente haciendo click con el ratón sobre los colores de interés en la propia foto o dibujo. La figura 4 ilustra este paso en nuestra aplicación.

Seguidamente, se descarga el mencionado formulario para dar paso a la ejecución del procedimiento que lo llamó y así poder continuar. Es ahora cuando realmente se compara pixel a pixel toda la imagen con cada uno de los colores seleccionados, generando una nueva imagen umbralizada en blanco y negro como muestra el ejemplo de la figura 5. Se trata de asignar el color blanco a todo pixel cuyo color sea parecido a cualquiera de los colores previamente seleccionados, de lo contrario se le asignará el color negro. En la selección de colores se ha admitido un margen de error de ±10% en cada componente RGB. Debido al elevado número de comprobaciones que requiere este proceso, especialmente si la selección se hace sobre un número de colores elevado, se ha incluido una barra de progreso que nos da una idea del tiempo restante para que el proceso termine.

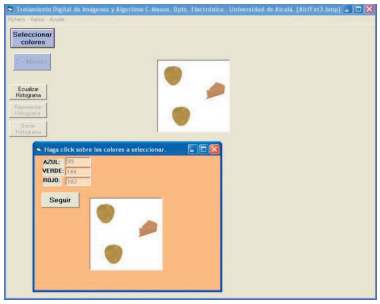

Figura 4. Proceso de selección de colores, previo a la umbralización de los objetos.

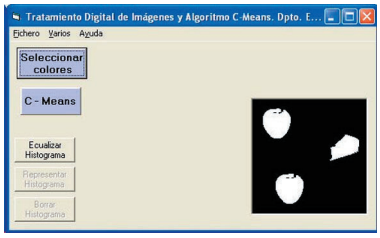

Figura 5. Imagen resultante del proceso de umbralizado que sigue a las operaciones de la figura 4.

Cuando termina el proceso de selección de colores obtenemos una imagen umbralizada en blanco y negro, que observaremos en pantalla, cuyo contenido será función de los colores que hayamos seleccionado, idealmente con los objetos de interés. Esta imagen es justamente la que necesitamos para aplicarle el algoritmo c-means, con objeto de separar o distinguir cada una de las agrupaciones de pixels que forman cada objeto. La figura 6 muestra un ejemplo de asignación de coordenadas de los prototipos. De este modo, una vez finalizado el clustering, conseguimos aislar cada objeto de los demás y la forma en la que se nos presenta dicho resultado en pantalla, es asignando a cada objeto un color distinto de los demás, justo como un proceso de etiquetado. Este color, por supuesto, no tiene por qué coincidir con el color que inicialmente tenía el objeto y sólo pretende distinguirlo de los demás. Adicionalmente, podemos ver la misma información en forma de coordenadas que corresponden a los centros de masas de cada uno de los objetos.

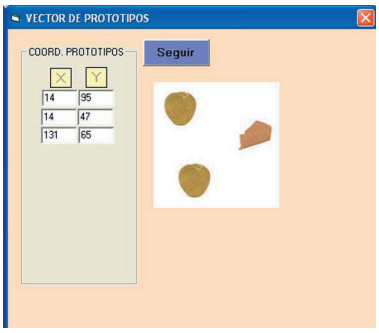

Figura 6. Ejemplo asignando las coordenadas de los prototipos.

Resumiendo brevemente el algoritmo c-means, en forma de pseudocódigo, su implementación sería la siguiente:

```
Hacer
  Recorrer filas Matriz_Patrones
    Recorrer columnas Matriz_Patrones
      Si se encuentra un patrón, es decir, si un pixel es >0 entonces
        Recorrer Vector de Prototipos
          Encontrar el prototipo más próximo al patrón y asignar al
          patrón un valor en función del prototipo más próximo.
        Siguiente prototipo
    Siguiente columna Matriz_Patrones
  Siguiente fila Matriz_Patrones
  Recalcular_Vector_de_Prototipos()
Hasta que Vector de Prototipos = Vector de Prototipos Anterior
```

Donde Matriz_Patrones representa la imagen umbralizada en función de los colores previamente seleccionados, y el Vector de Prototipos contiene las coordenadas de cada uno de los prototipos que hemos introducido(tantos como objetos queramos segmentar). A su vez, el algoritmo para recalcular el vector de prototipos sería:

```
Recalcular_Vector_de_Prototipos()
  Recorrer Vector de Prototipos
    Prototipo( i,0 ) = (1/Ni) · Σ x   (x∈xi)
    Prototipo( i,1 ) = (1/Ni) · Σ y   (y∈yi)
  Siguiente Prototipo
Fin
```

El presente algoritmo agrupa tantos conjuntos de pixels como número de prototipos introduzcamos, y va moviendo cada uno de los prototipos hasta colocarlo

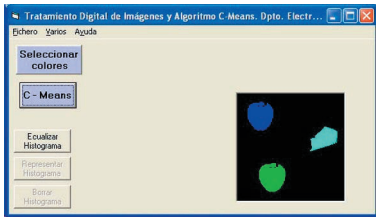

Figura 7. Imagen resultante del proceso de clustering

en el centro de masas de cada una de las agrupaciones u objetos. En nuestro programa, al hacer click en el botón C-Means se carga el formulario FormPrototipos que nos interroga, antes de nada, el número de prototipos que deseamos introducir en la imagen. Lógicamente, el número de prototipos debería ir en concordancia con el de objetos que deseemos segmentar. También se nos preguntará si deseamos ubicar los prototipos en forma numérica tecleando sus coordenadas, o si deseamos hacerlo gráficamente con el ratón. Terminamos haciendo click en el botón Seguir y comienza la ejecución del algoritmo.

Cuando termina c-means se nos mostrará una imagen con cada uno de los objetos coloreados con distinto tono, dependiendo del prototipo al que han quedado asignados, tal como muestra la figura 7. Si lo que deseamos es ver el valor de las coordenadas de los centros de masas de cada uno de los objetos segmentados, será suficiente volver a hacer click en C-Means y observar el valor de las coordenadas que ahora tienen los prototipos.

Conclusiones

Este trabajo comenta de una forma sencilla las operaciones de procesado de imágenes más típicas y describe cómo se han implementado en nustra aplicación software. Se ha realizado una descripción básica del algoritmo de agrupamiento c-means que también incluye nuestra aplicación, al que le sigue una detallada implementación del mismo en pseudocódigo, muy clara para poder ser implementada con cualquier lenguaje típico de programación. Algunos ejemplos ilustrativos muestran la utilización y resultados obtenidos con la aplicación que se desarrolló en Visual Basic.

Referencias

[1] Rafael C. González, Richard E. Woods, "Digital Image Processing", Addison-Wesley, Massachusetts, 1992.
[2] A. Domingo Ajenjo, "Tratamiento digital de imágenes", Anaya, Madrid, 1993.
[3] Jian Yu, "General c-means clustering model and its application", Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference .