

Cascade Classification K-Means

Dayan Bravo Fraga

March 2023

1 Instrucciones del Proyecto

El objetivo de este proyecto es el de hacer una clasificación en cascada de una imagen a color en formato CIE-Lab, utilizando únicamente los componentes ab de cada pixel.

La clasificación en cascada opera como sigue: en un primer nivel, cada pixel de la imagen se clasificará como perteneciente a dos clases. Esto se realizará utilizando el algoritmo de k-medias, con un valor de $K=2$. A los pixels clasificados en cada una de las dos clases, se clasificarán a su vez en otras dos posibles clases, utilizando también el algoritmo de k-medias. Se procede de esta manera con cada subclase encontrada hasta que no pueda ser subdividida. El algoritmo deberá producir un árbol binario en donde en cada iteración, a partir de las hojas del árbol, se pueda generar una imagen que sería una versión simplificada, en cuanto al número de colores, de la imagen original.

El algoritmo de K-medios se debe ejecutar primero utilizando la distancia Euclidiana como métrica, hasta que converja o hasta que el número de píxeles que cambia de clase es lo suficientemente pequeño. Una vez que ocurra eso, se deberá continuar ejecutando, pero utilizando la distancia de Mahalanobis como métrica.

Se debe entregar el programa funcionando, y un reporte en donde se reporte lo hecho y los resultados obtenidos.

2 Solution

En esta sección se describe el algoritmo utilizado para resolver el problema planteado. Al mismo tiempo, se explican los pasos que se deben seguir para poder ejecutar el programa. Así como un ejemplo de ejecución del programa utilizando una imagen de prueba.

2.1 Preparación del ambiente

Para poder ejecutar el programa, se debe tener instalado el compilador de C++ `g++` y la herramienta `make`. Además, se debe tener instalado el paquete `libopencv-dev` para poder utilizar la librería `OpenCV`.

Para instalar `libopencv-dev` se debe ejecutar el siguiente comando:

```
sudo apt-get install libopencv-dev
```

2.1.1 Compilación

Para compilar el programa, se debe ejecutar el siguiente comando:

```
make
```

2.1.2 Ejecución

El programa tiene una parameter opcional, que es el nombre de la imagen a utilizar. Si no se especifica el nombre de la imagen, se utilizará la imagen por defecto. Que es una imagen de prueba de 3x3.

Para ejecutar el programa, se debe ejecutar el siguiente comando:

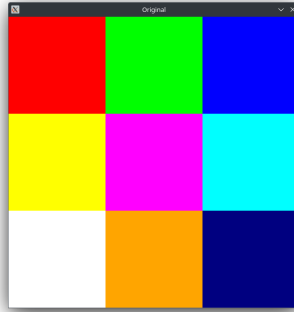


Figure 1: Imagen de prueba de 3x3.

```
./main
```

2.2 Algoritmo

1. Importar imagen a color en formato CIE-Lab.

Para esto se utiliza la siguiente función:

```
void ImageRepo::byName(
    Mat &image,
    Mat &imageOriginal,
    const string &name
) {
    // Read image.
    imageOriginal = imread(name);
    // Convert to float values.
    Mat imageFloat;
    imageOriginal.convertTo(imageFloat, CV_32FC3);
    // Normalize.
    imageFloat /= 255.0;
    // Convert to BGR2Lab.
    cvtColor(imageFloat, image, COLOR_BGR2Lab);
}
```

2. Se genera una máscara de la misma dimensión que la imagen, que contiene las clases a las que pertenece cada pixel. (Inicialmente, todos los píxeles pertenecen a la misma clase "0").

```
// init mask with zeros.
mask = Mat::zeros(image.size(), CV_8UC1);
```

3. Luego de dividir máscara en dos clases, se procede a dividir cada una de las clases en dos clases más.

3 Proof

3.1 Imagen de 3x3

Realizaremos una pequeña prueba con una imagen de 3x3, para ver que el algoritmo funciona correctamente.

La Figura 1 se va a utilizar para probar el algoritmo.

Al inicializar la máscara, todos los píxeles pertenecen a la misma clase, por lo que la máscara se inicializa con todos los píxeles con el valor 0. La máscara es la siguiente matriz:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Si graficamos la imagen con los valores promedio de todos los píxeles (de esa clase), obtenemos la siguiente imagen:

En una tabla lo podemos ver de la siguiente forma:

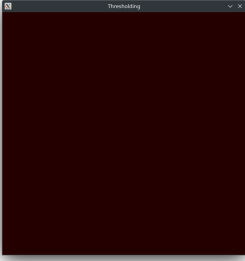
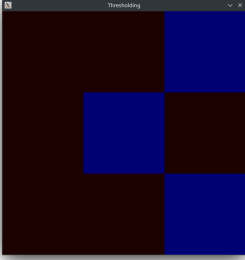
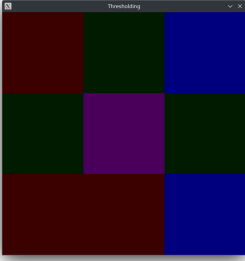
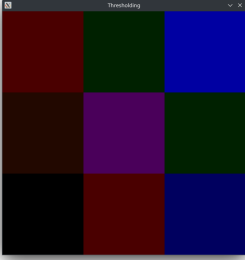

Mask	Image (L=0)
$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
$\begin{bmatrix} 2 & 2 & 1 \\ 2 & 1 & 2 \\ 2 & 2 & 1 \end{bmatrix}$	
$\begin{bmatrix} 5 & 6 & 3 \\ 6 & 4 & 6 \\ 5 & 5 & 3 \end{bmatrix}$	
$\begin{bmatrix} 9 & 11 & 7 \\ 12 & 4 & 11 \\ 10 & 9 & 8 \end{bmatrix}$	
$\begin{bmatrix} 13 & 15 & 7 \\ 12 & 4 & 16 \\ 10 & 14 & 8 \end{bmatrix}$	

Table 1: An example table.

Como podemos observar, la imagen tiene todos los píxeles con el mismo valor (promedio de todos los píxeles)

Ahora podemos dividir la clase 0 en 2 clases, luego se realiza la umbralización y se obtiene la siguiente máscara:

Si graficamos la imagen con los valores promedio de los píxeles de las nuevas clases (1 y 2), obtenemos la siguiente imagen:

Como podemos observar, la imagen se dividió en 2 clases, una con el valor promedio de los píxeles de la clase 1 y otra con el valor promedio de los píxeles de la clase 2.