

# Cascade Classification K-Means

Dayan Bravo Fraga

March 2023

## 1 Project instructions

The goal of this project is to perform a cascaded classification of a color image in CIE-Lab format, using only the ab components of each pixel. The cascaded classification operates as follows: in the first level, each pixel of the image will be classified as belonging to one of two classes. This will be done using the k-means algorithm, with a value of  $K=2$ . The pixels classified in each of the two classes will then be classified into two possible classes using the k-means algorithm as well. This process is repeated with each subclass found until it cannot be subdivided any further.

The algorithm should produce a binary tree where, in each iteration, starting from the leaves of the tree, a simplified version of the original image with fewer colors can be generated. The k-means algorithm should be executed first using the Euclidean distance as the metric until it converges or until the number of pixels that change class is small enough. Once this happens, the algorithm should continue running, but using the Mahalanobis distance as the metric.

You should deliver a functioning program and a report detailing what was done and the results obtained.

## 2 Procedure

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" ).
3. Luego de dividir mascarada 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.

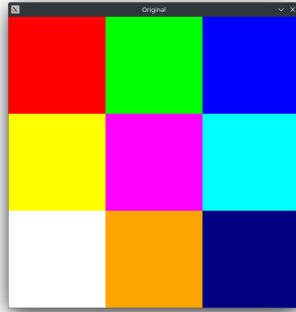


Figure 1: Imagen de prueba de 3x3.

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:

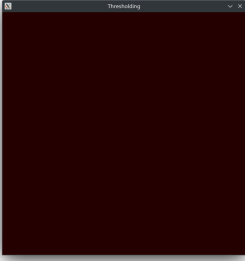
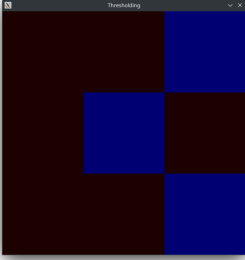
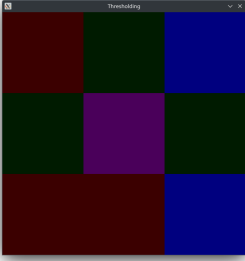
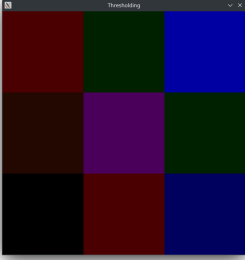

Mask	Image (L=0)
$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
$\begin{bmatrix} 2 & 2 & 1 \\ 2 & 1 & 2 \\ 2 & 2 & 1 \end{bmatrix}$	
$\begin{bmatrix} 5 & 6 & 3 \\ 6 & 4 & 6 \\ 5 & 5 & 3 \end{bmatrix}$	
$\begin{bmatrix} 9 & 11 & 7 \\ 12 & 4 & 11 \\ 10 & 9 & 8 \end{bmatrix}$	
$\begin{bmatrix}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

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.