LABORATORIO

Luis Ariza Ramos
Erick Barros Escorcia
Universidad del Magdalena, Ingeniería Electrónica,
Colombia
Luisarizaar@unimagdalena.edu.co
Erickbarrosde@unimagdalena.edu.co

Abstract **This laboratory practice carried out by means of the virtual and online tool "Colab", in phyton language a code proposed by the teacher, modify an RGB filter in order to separate various objects such as 6 balls of different colors in an image of type JPG.**

Resume Esta práctica de laboratorio realiza por medio de la herramienta virtual y online "Colab", en lenguaje phyton un codigo propuesto por el docente, modificar un filtro RGB con el fin de separar varios objetos como por ejemplo 6 pelotas de diferente color en una imagen de tipo JPG.

Keywords: RGB, filter, python

I. MARCO TEORICO

El término procesamiento digital de imágenes equivale a realizar operaciones sobre un conjunto de datos obtenidos de las imágenes, con el fin de mejorar la imagen para alguna aplicación en particular, ó para extraer algún tipo de información útil de ella [2]. En medicina por ejemplo, las técnicas de procesamiento realzan el contraste o codifican los niveles de intensidad en colores para facilitar la interpretación de las imágenes en rayos X [2], y de otras imágenes biomédicas. En arqueología, los métodos de procesamiento han servido para restaurar imágenes borrosas que eran los únicos registros existentes de piezas extrañas, perdidas o dañadas después de haber sido fotografiadas [1]. Para el sentido de la vista de los seres humanos, la extracción de la información de la imagen no tiene relación con los objetos del mundo que nos rodea [3]. Como ejemplo de tipos de información son, los momentos estadísticos y los coeficientes de la transformada de Fourier. Estos tipos de información se utilizan para el reconocimiento automático de caracteres, para la visión industrial mecanizada, entre otros [1].

Espacio de Color RGB Este modelo es un subespacio del espacio euclidiano conformado por el cubo unitario [1] mostrado en la Fig. 4.1. Los colores aparecen con sus componentes primarias de rojo, verde y azul. Los valores de R, G, y B se encuentran a lo largo de tres ejes. En otras palabras, en el eje del rojo, en el eje del verde y en el eje del azul se encuentran las intensidades de cada color. El cián está situado en el vértice del cubo en donde el color verde y el azul tienen su máximo valor, y el valor del rojo es cero; las coordenadas son (R, G, B) = (0, 1, 1). Análogamente, el magenta que es la combinación del rojo y el azul está situado en las coordenadas (R, G, B) = (1, 0, 1); y el amarillo (mezcla de verde con rojo) se situa en (R, G, B) = (1, 1, 0). El negro está posicionado en el origen del sistema y el blanco en el vértice opuesto al origen. La escala de grises se encuentra en la diagonal que va del negro al blanco, los colores restantes se encuentran dentro del cubo. Todos los valores de R, G y B están en el intervalo [0,1]. En el caso de imágenes digitales los valores de R, G y B son números enteros y van de 0 a 255, lo cual permite generar 16 777 216 colores.

La detección de la discontinuidad consiste en dividir una imagen basándose en los cambios bruscos del nivel de gris. Es particularmente importante porque proporciona información de los objetos de la imagen a otras tareas del procesamiento de imágenes como reconocimien-to e interpretación.[4] Los temas más importantes en la discontinuidad son: a) detección de puntos aislados, y b) detección de líneas y c) detección de bordes o con-tornos de una imagen. Aunque la detección de punto y línea son elementos de cualquier presentación de la segmentación de imágenes, la detección de bordes es la técnica más común para detectar discontinuidades significativas en el nivel de gris, debido a que son más frecuentes en las aplicaciones prácticas.Los métodos de extracción de bordes de

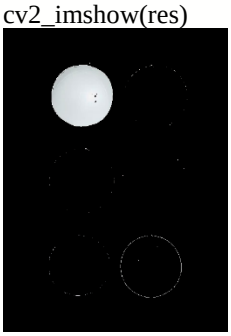
una imagen, se basan en la diferencia que experimenta una carac-terística en dos regiones adyacentes y que indican la existencia de un borde. A la vez los bordes pueden clasificarse por su anchura, ángulo de su pendiente de variación, y las coordenadas de su punto medio. En general, se identifican diferentes modelos de bordes o contornos: línea, tipo escalón, tipo rampa y tipo tejado. Las discontinuidades son detectadas usando derivadas de primer y segundo orden, enel caso de derivadas de primer orden se utiliza el operador gradiente, mientras que en derivadas de segundo orden se utiliza el opera-dor laplaciano.

II. DESARROLLO

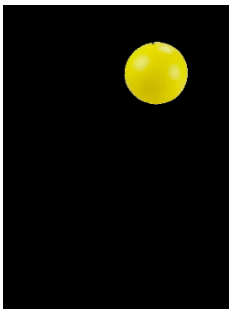
- En Colab de Google® cargar y mostrar la imagen enviada PELOTASLISASCOLORES.jpg
- Utilizando la detección de bordes, identifique solo los contornos de las 6 pelotas presentes en la imagen. Utilice diferentes métodos para la detección de bordes como:
  'Sobel', 'Prewitt' , 'Roberts' , 'log', 'zerocross','Canny'
  Y finalmente determinen a través de qué método se obtuvieron los mejores resultados.
- Realice un proceso de segmentación que permita visualizar cada una de las 6 pelotas presentes en la imagen, de manera independiente, como se muestra a continuación. Utilice diferentes espacios de color para alcanzar este objetivo y finalmente determinen a través de qué método se obtuvieron los mejores resultados.

```
# PROCESAMIENTO DE CONTENIDO MULTIMEDIA
# Luis Ariza - Erick Barros
#Importando librerias
from PIL import Image
from matplotlib import image
from matplotlib import pyplot
import matplotlib.pyplot as plt
import cv2
import numpy as np
from google.colab.patches import cv2_imshow

#Leer Imágenes
img2=cv2.imread('PELOTASLISAS-COLORES.jpg')
rgb2=cv2.cvtColor(img2,cv2.COLOR_BGR2RGB)
hsv=cv2.cvtColor(img2, cv2.COLOR_BGR2HSV)
cv2_imshow(img2)
```

```
#White
HSV= cv2.cvtColor(img2, cv2.COLOR_BGR2HSV)
lower_W= np.array ([0,0,190])
upper_W= np.array ([175,20,245])
mask= cv2.inRange (HSV, lower_W, upper_W)
res= cv2.bitwise_and(img2,img2, mask=mask)
cv2_imshow(res)
```

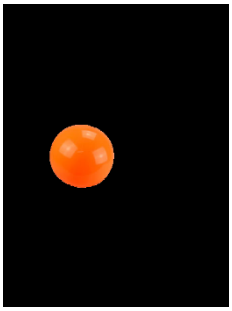


```
#Yellow

HSV= cv2.cvtColor(img2, cv2.COLOR_BGR2HSV)
lower_Y= np.array ([18,52,180])
upper_Y= np.array ([50,255,255])
mask= cv2.inRange (HSV, lower_Y, upper_Y)
res= cv2.bitwise_and(img2,img2, mask=mask)
cv2_imshow(res)
```



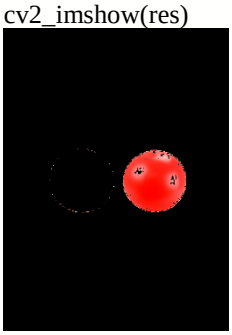
#Orange

```
HSV= cv2.cvtColor(img2, cv2.COLOR_BGR2HSV)
lower_O= np.array ([6,100,35])
upper_O= np.array ([22,255,255])
mask= cv2.inRange (HSV, lower_O, upper_O)
res= cv2.bitwise_and (img2, img2, mask=mask)
cv2_imshow(res)
```



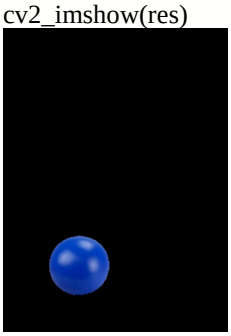
#Red

```
HSV= cv2.cvtColor(img2, cv2.COLOR_BGR2HSV)
lower_R= np.array ([0,20,60])
upper_R= np.array ([5,255,255])
mask= cv2.inRange (HSV, lower_R, upper_R)
res= cv2.bitwise_and(img2,img2, mask=mask)
cv2_imshow(res)
```



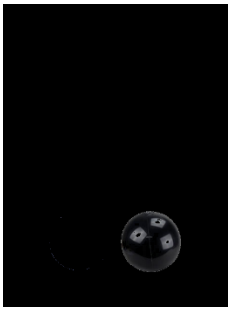
#Blue

```
HSV= cv2.cvtColor(img2, cv2.COLOR_BGR2HSV)
lower_B= np.array ([98,95,20])
upper_B= np.array ([150,255,255])
mask= cv2.inRange (HSV, lower_B, upper_B)
res= cv2.bitwise_and(img2,img2, mask=mask)
cv2_imshow(res)
```



#Black

```
HSV= cv2.cvtColor(img2, cv2.COLOR_BGR2HSV)
lower_BL= np.array ([0,0,0])
upper_BL= np.array ([180,120,150])
mask= cv2.inRange (HSV, lower_BL, upper_BL)
res= cv2.bitwise_and(img2,img2, mask=mask)
cv2_imshow(res)
```



III. CONCLUSIONES

El filtrado de imágenes en RGB es una técnica muy útil para procesar imágenes y mejorar la calidad visual. El filtrado en RGB implica la aplicación de un filtro a cada canal de color (rojo, verde y azul) por separado. El filtro puede ser un filtro como la eliminación de ruido, la mejora de bordes, la mejora de contraste y la eliminación de imperfecciones en la imagen. Los resultados del filtrado en RGB dependen del tipo de filtro utilizado y de los parámetros de filtrado que se ajusten.

En general, el filtrado en RGB es una técnica efectiva para mejorar la calidad visual de las imágenes y puede ser utilizado en una amplia variedad de aplicaciones, desde la fotografía hasta la visión artificial y la robótica.

IV. BIBLIOGRAFIA

[1] R. C. Gonzalez and R. E. Woods: Digital Image Processing (Prentice-Hall, New Jersey, 2002) Cap. 1, p. 6; Cap. 2, p.50; Cap. 6, pp. 284-299.
[2]J. Y. Hardeberg: Acquisition and Reproduction of Color Images (PWS Publishing Company, Boston, 2001) Cap. 2, p. 15.
[3] E. L. Hall: Computer Image Processing and Recognition (Academic Press, New York, 1979) Cap. 2, p. 8.
[4] W. K. Pratt: Digital Image Processing (Jhon Wiley & Sons, Inc., New York, 1991) Cap. 2, p. 93; Cap. 10, p. 310