UNIVERSIDAD NACIONAL DE TRUJILLO

Facultad de ingeniería
Programa de ingeniería mecatrónica

SEMANA 8 – PRÁCTICA – PROCESAMIENTO DE COLOR

PROCESAMIENTO DIGITAL DE SEÑALES E IMÁGENES

ESTUDIANTE(S) :
CORTEZ GOMEZ BRAJAN LEONEL
ESPINOZA LEÓN KARL ALEJANDRO
GUTIÉRREZ GUTIÉRREZ ITALO AARÓN

DOCENTE :
MS. ING. EMERSON MÁXIMO ASTO RODRIGUEZ

CICLO :
2023 - II

Trujillo, Perú
2024

Objetivos:

- Familiarizarse con las operaciones de suavizado y enfatizado en el procesamiento de imágenes.
- Comprender y aplicar algoritmos para la implementación de filtros frecuenciales en imágenes digitales.

Instrucciones:

1. Revise e implemente la interfaz basándose en el snippet siguiente.

```
import tkinter as tk
from PIL import Image, ImageTk
import cv2
import numpy as np

def process_frame():
    _, frame = cap.read()

    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    lower = np.array([0, 90, 50])
    upper = np.array([60, 220, 180])

    mask = cv2.inRange(hsv, lower, upper)
    res = cv2.bitwise_and(frame, frame, mask=mask)

    update_label(frame, label_frame)
    update_label(mask, label_mask)
    update_label(res, label_result)

    ventana.after(20, process_frame)

def update_label(image, label):
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    image = resize_image(image, 300)
    image = Image.fromarray(image)
    image_tk = ImageTk.PhotoImage(image=image)
    label.config(image=image_tk)
    label.image = image_tk

def resize_image(image, max_width):
    original_height, original_width, _ = image.shape
    ratio = max_width / original_width
    height = int(original_height * ratio)
    return cv2.resize(image, (max_width, height))

ventana = tk.Tk()
ventana.title("Real-time Image Processing with Tkinter")

cap = cv2.VideoCapture(0)

label_frame = tk.Label(ventana)
label_frame.grid(row=0, column=0, padx=10, pady=10)

label_mask = tk.Label(ventana)
label_mask.grid(row=0, column=1, padx=10, pady=10)

label_result = tk.Label(ventana)
label_result.grid(row=0, column=2, padx=10, pady=10)

process_frame()

ventana.mainloop()

cap.release()
cv2.destroyAllWindows()
```

2. Implemente un algoritmo que le permita realizar la operación de balance de blancos.

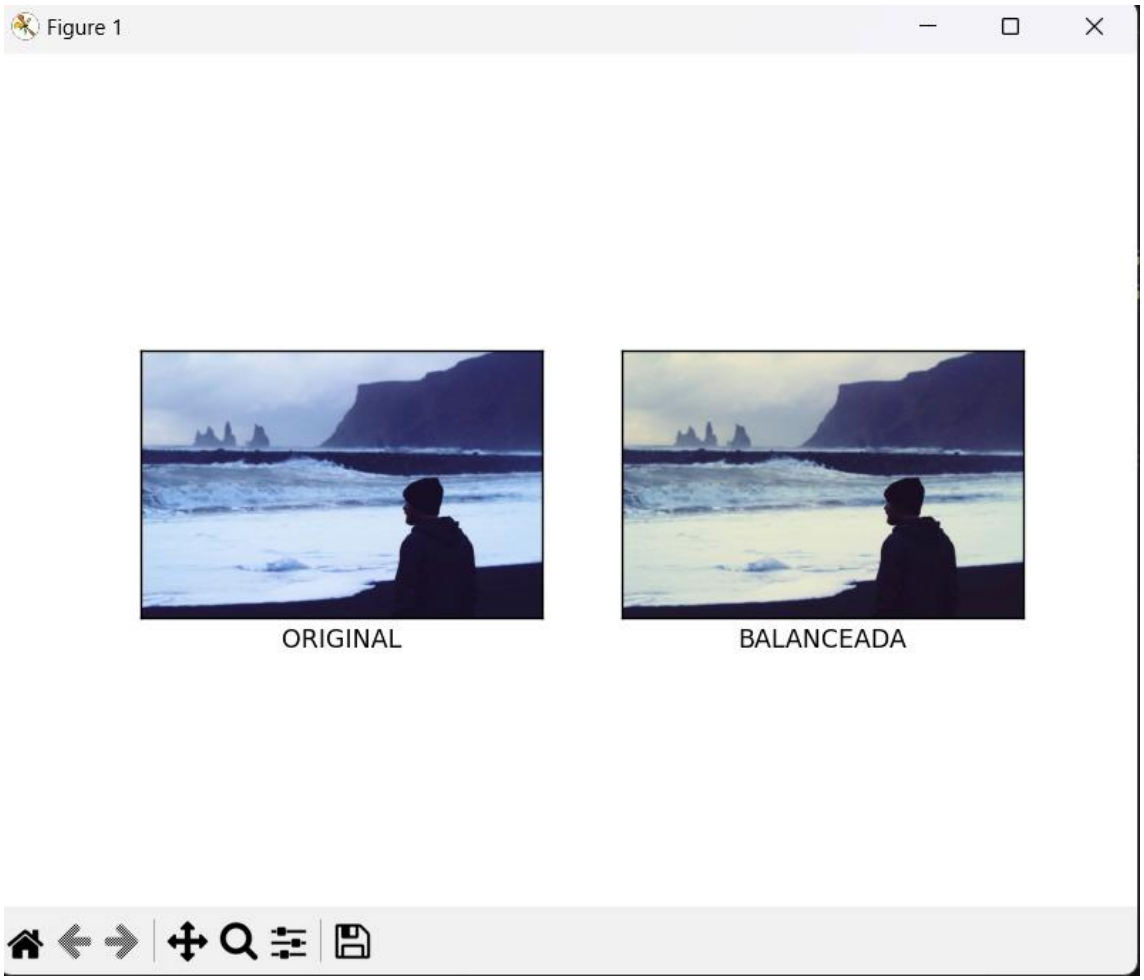
Código implementado:

```
import numpy as np
import matplotlib.pyplot as plt
import cv2

img = cv2.imread('balance.jpg')
img_RGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
img_YUV = cv2.cvtColor(img, cv2.COLOR_BGR2YUV)
avg_U = np.average(img_YUV[:, :, 1])
avg_V = np.average(img_YUV[:, :, 2])
img_YUV[:, :, 1] = img_YUV[:, :, 1] - ((avg_U - 128) * (img_YUV[:, :, 0] / 255.0) * 1.2)
img_YUV[:, :, 2] = img_YUV[:, :, 2] - ((avg_V - 128) * (img_YUV[:, :, 0] / 255.0) * 1.2)
img_balanceada = cv2.cvtColor(img_YUV, cv2.COLOR_YUV2RGB)

plt.subplot(1,2,1)
plt.xlabel('ORIGINAL')
plt.xticks([]), plt.yticks([])
plt.imshow(img_RGB)
plt.subplot(1,2,2)
plt.xlabel('BALANCEADA')
plt.xticks([]), plt.yticks([])
plt.imshow(img_balanceada)
plt.show()
```

Resultado de la ejecución:



3. Implemente una interfaz que permita ingresar los 3 valores hsv (o cualquier otro espacio de color) y segmentarlos (Agregar 3 TextInput/sliders y un botón).

Código implementado:

```python
import tkinter as tk
from PIL import Image, ImageTk
import cv2
import numpy as np

def process_frame():
    _, frame = cap.read()

    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    lower = np.array([hue_lower.get(), saturation_lower.get(), value_lower.get()])
    upper = np.array([hue_upper.get(), saturation_upper.get(), value_upper.get()])

    mask = cv2.inRange(hsv, lower, upper)
    res = cv2.bitwise_and(frame, frame, mask=mask)

    update_label(frame, label_frame)
    update_label(mask, label_mask)
    update_label(res, label_result)

    ventana.after(20, process_frame)

def update_label(image, label):
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    image = resize_image(image, 350)
    image = Image.fromarray(image)
    image_tk = ImageTk.PhotoImage(image=image)
    label.config(image=image_tk)
    label.image = image_tk

def resize_image(image, max_width):
    original_height, original_width, _ = image.shape
    ratio = max_width / original_width
    height = int(original_height * ratio)
    return cv2.resize(image, (max_width, height))
```

```
ventana = tk.Tk()
ventana.title("Real-time Image Processing with Tkinter")

cap = cv2.VideoCapture(0)

label_frame = tk.Label(ventana)
label_frame.grid(row=0, column=0, padx=10, pady=10)

label_mask = tk.Label(ventana)
label_mask.grid(row=0, column=1, padx=10, pady=10)

label_result = tk.Label(ventana)
label_result.grid(row=0, column=2, padx=10, pady=10)

# Agregar tres sliders para ajustar los valores HSV
hue_lower = tk.Scale(ventana, from_=0, to=180, orient=tk.HORIZONTAL, label="Hue Lower")
hue_lower.set(0)
hue_lower.grid(row=1, column=0, padx=10, pady=10)

saturation_lower = tk.Scale(ventana, from_=0, to=255, orient=tk.HORIZONTAL, label="Saturation Lower")
saturation_lower.set(90)
saturation_lower.grid(row=1, column=1, padx=10, pady=10)

value_lower = tk.Scale(ventana, from_=0, to=255, orient=tk.HORIZONTAL, label="Value Lower")
value_lower.set(50)
value_lower.grid(row=1, column=2, padx=10, pady=10)

hue_upper = tk.Scale(ventana, from_=0, to=180, orient=tk.HORIZONTAL, label="Hue Upper")
hue_upper.set(60)
hue_upper.grid(row=2, column=0, padx=10, pady=10)

saturation_upper = tk.Scale(ventana, from_=0, to=255, orient=tk.HORIZONTAL, label="Saturation Upper")
saturation_upper.set(220)
saturation_upper.grid(row=2, column=1, padx=10, pady=10)
```

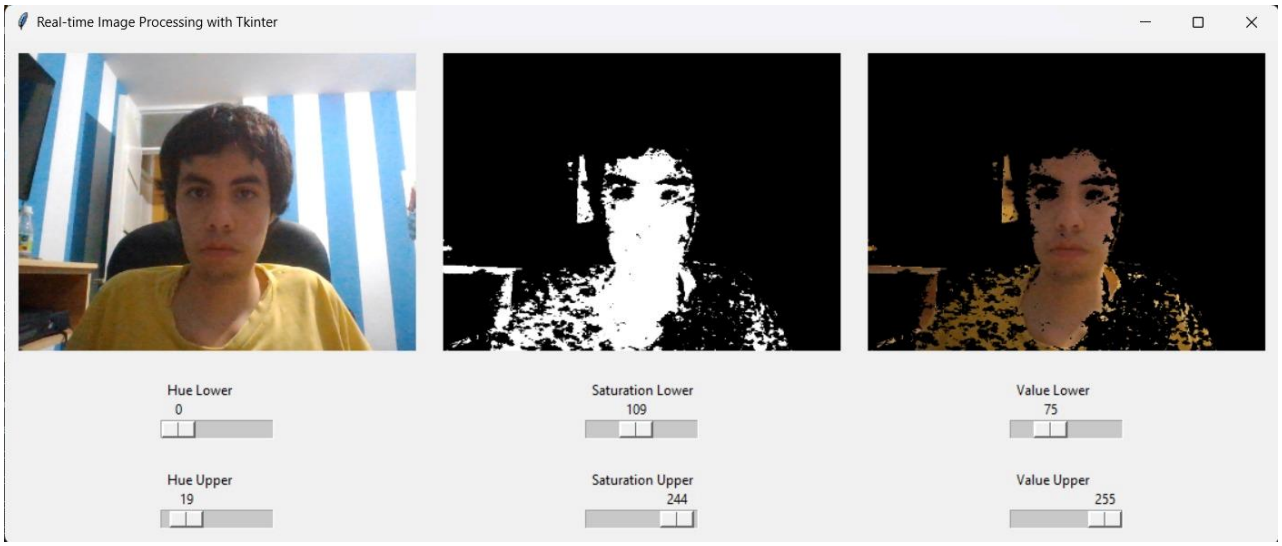
```
value_upper = tk.Scale(ventana, from_=0, to=255, orient=tk.HORIZONTAL, label="Value Upper")
value_upper.set(180)
value_upper.grid(row=2, column=2, padx=10, pady=10)

process_frame()

ventana.mainloop()

cap.release()
cv2.destroyAllWindows()
```

Resultado de la ejecución:


Real-time Image Processing with Tkinter
Hue Lower
0
Saturation Lower
109
Value Lower
75
Hue Upper
19
Saturation Upper
244
Value Upper
255

BIBLIOGRAFÍA

Oppenheim, A. V., Willsky, A. S., & Young, I. T. (1983). *Signals and systems*. Englewood Cliffs, N.J: Prentice-Hall.

Kamen, Edward W., y Bonnie S. Heck. (2008). Fundamentos de señales y sistemas usando la Web y MATLAB® PEARSON EDUCACIÓN, México, ISBN: 978-970-26-1187-5