UNIVERSIDAD DEL BÍO-BÍO

13 Umbralización RGB

Tutorial Procesamiento de Imagen con webcam

Estudiantes Practicantes:
Luis Pereira

Profesor:
Luis Vera

Laboratorio CIMUBB

2023-2

Umbralización de imagen a color

En este tutorial, presentamos un programa de procesamiento de imágenes que utiliza la umbralización RGB y la webcam. El objetivo es capturar imágenes en tiempo real desde la cámara web y aplicar la umbralización RGB para destacar áreas de interés.

Importación de bibliotecas:

Importamos las bibliotecas necesarias para el funcionamiento del programa, como tkinter para la interfaz gráfica, OpenCV para el procesamiento de imágenes y NumPy para operaciones numéricas.

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

Interfaz de Usuario:

Creamos una ventana principal usando Tkinter con un título descriptivo y un tamaño predeterminado que el usuario no puede cambiar.

```
# Creamos la ventana principal con titu
ventana = tk.Tk()
ventana.geometry("1400x730")
ventana.resizable(0, 0)
ventana.title("Umbralización RGB")

```

Iniciar la cámara:

Implementamos una función para iniciar la cámara web cuando el usuario hace clic en el botón "Iniciar cámara". La cámara captura imágenes en tiempo real y las muestra en un cuadro en la ventana.

```
def camara():
    global capture
    capture = cv2.VideoCapture(0)
    iniciar()

# Función para mostrar la imagen de la cámara
def iniciar():
    global capture
    if capture is not None:
        ret, frame = capture.read()
        if ret == True:
            frame = imutils.resize(frame, width=311)
            frame = imutils.resize(frame, height=241)
            ImagenCamara = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            im = Image.fromarray(ImagenCamara)
            img = ImageTk.PhotoImage(image=im)
            LImagen.configure(image=img)
            LImagen.image = img
            LImagen.after(1, iniciar)
        else:
            LImagen.image = ""
            capture.release()
```

Capturar una Foto:

Otra función es **Capturar**() permite al usuario capturar una foto haciendo clic en el botón "Tomar foto". La imagen capturada se muestra en un cuadro adyacente.

```
def Capturar():
    global Captura
    camara = capture
    return_value, image = camara.read()
    frame = imutils.resize(image, width=301)
    frame = imutils.resize(frame, height=221)
    Captura = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    im = Image.fromarray(Captura)
    img = ImageTk.PhotoImage(image=im)
    LImagenROI.configure(image=img)
    LImagenROI.image = img
```

Umbralización de Canales:

En el código proporcionado, se encuentran cuatro botones en la interfaz gráfica, cada uno de los cuales se utiliza para desencadenar una función llamada **umbralizar(n).** Esta función se encarga de procesar y mostrar diferentes aspectos de una imagen. Los botones están etiquetados como "Umbralizar en R", "Umbralizar en G", "Umbralizar en B" y "Imagen Fusionada", y cada uno tiene un propósito específico. El botón "Umbralizar en R" (BRojo) activa la función **umbralizar(n)** con n establecido en 1. Esta función procesa el canal de color rojo (R) de la imagen capturada y muestra la imagen umbralizada resultante en el área designada para el canal rojo en la interfaz gráfica. El botón "Umbralizar en G" (BVerde) activa la función **umbralizar(n)** con n establecido en 2. La función **umbralizar(n)** procesa el canal de color verde (G) de la imagen capturada y muestra la imagen umbralizada correspondiente en el área destinada al canal verde en la interfaz gráfica. El botón "Umbralizar en B" (BAzul) utiliza la función **umbralizar(n)** con n configurado en 3. La función umbralizar(n) procesa el canal de color azul (B) de la imagen capturada y muestra la imagen umbralizada resultante en el área asignada para el canal azul en la interfaz gráfica.

Por último, el botón "Imagen Fusionada" (BUmbralizar) activa la función **umbralizar(n)** con n establecido en 4. En este caso, la función **umbralizar(n)** toma los valores umbral definidos por el usuario en los canales de color R, G y B, y fusiona los tres canales para crear una imagen final umbralizada. Luego, muestra esta imagen compuesta en el área designada para la imagen fusionada en la interfaz gráfica.

```python
def umbralizar(n):
    canal_r, canal_g, canal_b = cv2.split(Captura)
    umbral_r_min = int(SRedI.get())
    umbral_g_min = int(SGreenI.get())
    umbral_b_min = int(SBlueI.get())
    umbral_r_max = int(SRedD.get())
    umbral_g_max = int(SGreenD.get())
    umbral_b_max = int(SBlueD.get())

    umbralizado_r = cv2.threshold(canal_r, umbral_r_min, umbral_r_max, cv2.THRESH_BINARY)[1]
    umbralizado_g = cv2.threshold(canal_g, umbral_g_min, umbral_g_max, cv2.THRESH_BINARY)[1]
    umbralizado_b = cv2.threshold(canal_b, umbral_b_min, umbral_b_max, cv2.THRESH_BINARY)[1]
    imagen_umbralizada = cv2.merge((umbralizado_r, umbralizado_g, umbralizado_b))

    if n == 1:
        im = Image.fromarray(umbralizado_r)
        img = ImageTk.PhotoImage(image=im)
        ImagenRed.configure(image=img)
        ImagenRed.image = img
    elif n == 2:
        im = Image.fromarray(umbralizado_g)
        img = ImageTk.PhotoImage(image=im)
        ImagenGreen.configure(image=img)
        ImagenGreen.image = img
    elif n == 3:
        im = Image.fromarray(umbralizado_b)
        img = ImageTk.PhotoImage(image=im)
        ImagenBlue.configure(image=img)
        ImagenBlue.image = img
    elif n == 4:
        im = Image.fromarray(imagen_umbralizada)
        img = ImageTk.PhotoImage(image=im)
        ImagenUmbra.configure(image=img)
        ImagenUmbra.image = img
```

Elementos de Interfaz:

Se crean escalas deslizantes (Sliders) para permitir al usuario seleccionar los valores de umbral mínimo y máximo para los canales RGB.

Se añaden etiquetas para indicar la letra inicial de cada color (R, G, B).

Se disponen botones para activar las funciones, como iniciar la cámara, tomar una foto y aplicar la umbralización.

```
SRedD = tk.Scale(ventana, from_=1, to=255, orient='horizontal')
SRedD.set(255)
SRedD.place(x=220, y=630)
SGreenD = tk.Scale(ventana, from_=1, to=255, orient='horizontal')
SGreenD.set(255)
SGreenD.place(x=550, y=630)
SBlueD = tk.Scale(ventana, from_=1, to=255, orient='horizontal')
SBlueD.set(255)
SBlueD.place(x=900, y=630)

SRedI = tk.Scale(ventana, from_=1, to=255, orient='horizontal')
SRedI.place(x=80, y=630)
SGreenI = tk.Scale(ventana, from_=1, to=255, orient='horizontal')
SGreenI.place(x=410, y=630)
SBlueI = tk.Scale(ventana, from_=1, to=255, orient='horizontal')
SBlueI.place(x=760, y=630)
```

Visualización de Resultados:

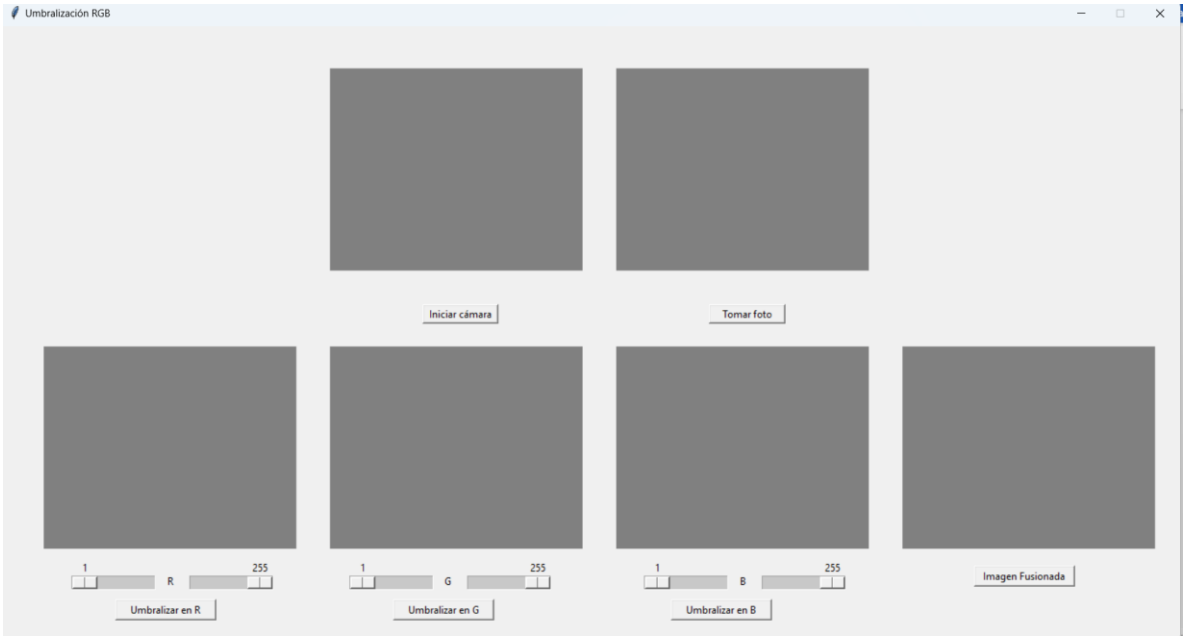
Se han incorporado cuadros grises para mostrar las imágenes capturadas, umbralizadas y fusionadas. Estos cuadros se actualizan en tiempo real a medida que se aplican las funciones.

```
LImagen = tk.Label(ventana, background="gray")
LImagen.place(x=390, y=50, width=300, height=240)
LImagenROI = tk.Label(ventana, background="gray")
LImagenROI.place(x=730, y=50, width=300, height=240)

ImagenRed = tk.Label(ventana, background="gray")
ImagenRed.place(x=50, y=380, width=300, height=240)
ImagenGreen = tk.Label(ventana, background="gray")
ImagenGreen.place(x=390, y=380, width=300, height=240)
ImagenBlue = tk.Label(ventana, background="gray")
ImagenBlue.place(x=730, y=380, width=300, height=240)
ImagenUmbra = tk.Label(ventana, background="gray")
ImagenUmbra.place(x=1070, y=380, width=300, height=240)
```

El programa proporciona una interfaz interactiva para capturar imágenes de la webcam y aplicar la umbralización RGB de manera sencilla. Es un recurso valioso para la educación y la experimentación con el procesamiento de imágenes.

Resultado Final:



En funcionamiento…