UNIVERSIDAD DEL BÍO-BÍO

10 Tomar foto usando webcam y transformarla a escala de grises

Tutorial Procesamiento de Imagen con webcam

Estudiantes Practicantes:
Luis Pereira

Profesor:
Luis Vera

Laboratorio CIMUBB

2023-2

Importación de las bibliotecas

La aplicación desarrollada en Python combina las capacidades de diversas bibliotecas, como tkinter, OpenCV (cv2), Pillow (PIL), imutils y os, para proporcionar una interfaz amigable para la captura de imágenes desde una cámara web y su posterior almacenamiento en el sistema de archivos del usuario.

```python
import tkinter as tk
from tkinter import *
from tkinter import filedialog, messagebox
from PIL import Image
from PIL import ImageTk
import imutils
import cv2
import os
```

Creación de la Ventana

Iniciamos creando una ventana con la biblioteca tkinter. Definimos su tamaño, bloqueamos la posibilidad de que el usuario pueda modificar el tamaño y le añadimos un título significativo:

```python
# Crear ventana, definir tamaño y título
ventana = tk.Tk()
ventana.geometry("740x460")
ventana.resizable(0, 0)
ventana.title("Tomar foto webcam")
```

Funciones para Controlar la Webcam

Luego, definimos las funciones que nos permitirán controlar la cámara web. La función camara() inicia la cámara web utilizando OpenCV:

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

def iniciar():
    global capture
    if capture is not None:
        ret, frame = capture.read()
        if ret:
            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()
```

La función **iniciar**() muestra la imagen de la cámara web en una ventana de tkinter. Utiliza la biblioteca imutils para redimensionar la imagen y Pillow (PIL) para convertirla en un formato que tkinter puede mostrar.

Capturar una Foto en Escala de Grises

Creamos la función **CapturarG**() para capturar la foto, en la variable CapturaG le asignamos el valor de un frame de nuestra webcam y le ponemos la opción **cv2.COLOR_BGR2GRAY** para que nos guarde la imagen en escala de grises.

```
def CapturarG():
    global Captura
    cam = capture
    ret, image = cam.read()
    frame = imutils.resize(image, width=301)
    frame = imutils.resize(frame, height=221)
    Captura = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    im = Image.fromarray(Captura)
    img = ImageTk.PhotoImage(image=im)
    LImagenROI.configure(image=img)
    LImagenROI.image = img
```

Capturar Fotos con Temporización

La función **capturarG_con_temporizacion**() permite capturar fotos con un intervalo de tiempo predefinido. Esta función muestra una advertencia la primera vez que se ejecuta, indicando que se están guardando fotos en un ciclo infinito:

```
def capturarG_con_temporizacion():
    global ejecutar_solo_una_vez
    if ejecutar_solo_una_vez:···
    ejecutar_solo_una_vez = False

    global Captura
    cam = capture
    ret, image = cam.read()
    frame = imutils.resize(image, width=301)
    frame = imutils.resize(frame, height=221)
    Captura = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    im = Image.fromarray(Captura)
    img = ImageTk.PhotoImage(image=im)
    LImagenROI.configure(image=img)
    LImagenROI.image = img
    # Programa la próxima captura
    guardar_imagen()
    ventana.after(1000 * int(numeroTemp.get()), capturarG_con_temporizacion)
```

Interfaz Gráfica

Luego, creamos botones para que el usuario pueda iniciar la cámara y tomar una foto. También se agregan cuadros de imagen en escala de grises para mostrar la imagen de la cámara web y la foto capturada:

```
# Botón para seleccionar carpeta de destino
carpeta_seleccionar_button = tk.Button(ventana, text="Seleccionar Carpeta", command=seleccionar_carpeta)
carpeta_seleccionar_button.pack(pady=10)
carpeta_seleccionar_button.place(x=570, y=380)

# Botón para guardar la imagen
guardar_boton = tk.Button(ventana, text="Guardar Imagen", command=guardar_imagen, state=tk.DISABLED)
guardar_boton.pack(pady=10)
guardar_boton.place(x=580, y=420)

# Botón para iniciar la cámara web
BCamara = tk.Button(ventana, text="Iniciar cámara", command=camara)
BCamara.place(x=150, y=330, width=90, height=23)

# Botón para capturar una foto única
BCapturarUnica = tk.Button(ventana, text="Tomar foto única", command=CapturarG)
BCapturarUnica.place(x=500, y=305, width=100, height=23)

# Botón para capturar imágenes con temporización
BCapturarTemporizada = tk.Button(ventana, text="Tomar foto temporizada", command=capturarG_con_temporizacion)
BCapturarTemporizada.place(x=500, y=345, width=150, height=23)
```

Selección de Carpeta y Guardado

El usuario puede seleccionar una carpeta para guardar las fotos. Se muestra una advertencia al usuario la primera vez que intenta capturar una foto con temporización para recordarle que se está ejecutando un ciclo infinito:

```
def seleccionar_carpeta():
    global ruta_destino
    ruta_destino = filedialog.askdirectory()
    if ruta_destino:
        guardar_boton.config(state=tk.NORMAL)
        carpeta_seleccionada_label.config(text=f"Carpeta seleccionada: {ruta_destino}")

# Función para guardar una imagen en la carpeta seleccionada
def guardar_imagen():
    global Captura, ruta_destino
    if Captura is not None and ruta_destino:
        if not os.path.exists(ruta_destino):
            os.makedirs(ruta_destino)

        file_name = entrada_nombre_archivo.get() + ".png"
        ruta_guardar = os.path.join(ruta_destino, file_name)

        contador = 1
        while os.path.exists(ruta_guardar):
            contador += 1
            file_name = entrada_nombre_archivo.get() + f"({contador})" + ".png"
            ruta_guardar = os.path.join(ruta_destino, file_name)

        imagen_pil = Image.fromarray(Captura)
        imagen_pil.save(ruta_guardar)

        print(f"Fotografía guardada como {ruta_guardar}")
```

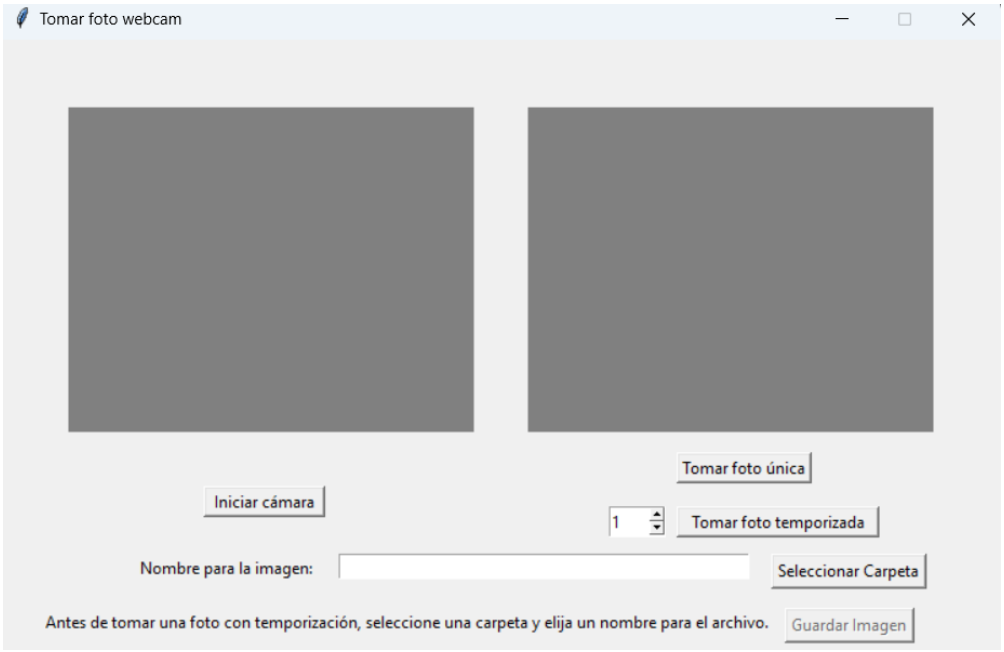
Finalmente, utilizamos **ventana.mainloop**() para asegurarnos de que todos los elementos de la interfaz gráfica se muestren correctamente. Con estas funcionalidades, los usuarios pueden tomar fotos de su cámara web y guardarlas en escala de grises, tanto de forma única como con intervalos de tiempo programados.

```
# Etiqueta y entrada para el nombre del archivo
label_nombre_archivo = tk.Label(ventana, text="Nombre para la imagen: ")
label_nombre_archivo.place(x=100, y=380)
entrada_nombre_archivo = tk.Entry(ventana, width=50)
entrada_nombre_archivo.place(x=250, y=380)

# SpinBox para seleccionar los segundos de la temporización
numeroTemp = tk.Spinbox(ventana, from_=1, to=60)
numeroTemp.place(x=450, y=345, width=42, height=23)

# Iniciar la interfaz gráfica
ventana.mainloop()
```

Resultado Final



Al tomar una foto se verá así, notar que no es necesario ponerle a la imagen la extensión ya que el programa por defecto guardara la captura en .png.