UNIVERSIDAD DEL BÍO-BÍO

15 Análisis de manchas

Tutorial Procesamiento de Imagen con webcam

Estudiantes Practicantes:
Luis Pereira

Profesor:
Luis Vera

Laboratorio CIMUBB

2023-2

Análisis de manchas blancas y negras

El presente tutorial describe el desarrollo de un proyecto de análisis de manchas en imágenes. El objetivo del proyecto es tomar una captura a color y, dependiendo de su umbralización, analizar cuántas manchas hay y qué porcentaje de la imagen ocupan. A lo largo de este informe, se describirá el código utilizado, las funciones implementadas y los pasos seguidos para lograr este análisis.

Importación de bibliotecas:

tkinter para la interfaz gráfica., PIL (Python Imaging Library) para el manejo de imágenes., imutils para procesamiento de imágenes., cv2 (OpenCV) para captura y procesamiento de imágenes.

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

Creación de la Ventana

Se creó una ventana principal con una resolución de 1320x470 píxeles, y se desactivó la capacidad de cambiar su tamaño. La ventana se tituló "Análisis de Manchas".

```
ventana = tk.Tk()
ventana.geometry("1320x470")
ventana.resizable(0, 0)
ventana.title("Análisis de Manchas")
```

Funciones de la Cámara Web

Se implementaron funciones para la cámara web. La función camara() permite iniciar la cámara, y la función **iniciar**() muestra la captura en tiempo real.

```python
# Funciones de la cámara web
def camara():
    global capture
    # Iniciar la cámara
    capture = cv2.VideoCapture(0)
    iniciar()

def iniciar():
    global capture
    # Mostrar la captura en tiempo real
    if capture is not None:
        ret, frame = capture.read()
        if ret:
            # Redimensionar el marco para mostrarlo en la ventana
            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()
```

Captura de Imagen

Se implementó la función **Capturar()**, que permite tomar una foto utilizando la cámara y mostrarla en escala de grises.

```python
def Capturar():
    global CapturaG
    camara = capture
    return_value, image = camara.read()
    frame = imutils.resize(image, width=301)
    frame = imutils.resize(frame, height=221)
    CapturaG = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    imG = Image.fromarray(CapturaG)
    imgG = ImageTk.PhotoImage(image=imG)
    LImagenROI.configure(image=imgG)
    LImagenROI.image = imgG
```

Umbralización de Escala de Grises

Se creó una función **umbralizar_escala_grises()** que permite realizar la umbralización en escala de grises. El usuario puede seleccionar un valor entre 0 y 255, y la función umbraliza la imagen de acuerdo a ese valor.

```python
def umbralizar_escala_grises():
    global bin_image
    valor = int(numeroUmbra.get())
    ret, bin_image = cv2.threshold(CapturaG, valor, 255, cv2.THRESH_BINARY)
    Umbral = Image.fromarray(bin_image)
    Umbral = ImageTk.PhotoImage(image=Umbral)
    ImagenUmbra.configure(image=Umbral)
    ImagenUmbra.image = Umbral
```

Análisis de Manchas

La función **contar_manchas_negras**() se encarga de analizar una imagen que ha sido umbralizada previamente y contar las manchas negras presentes en ella. Esto se logra a través de los siguientes pasos:

1) Invierte la imagen binarizada imagen_invertida utilizando la función **cv2.bitwise_not**. Esta inversión es necesaria porque asume que las manchas son áreas negras sobre un fondo blanco.
2) Calcula el número de píxeles con manchas negras utilizando **cv2.countNonZero(imagen_invertida)**. Esto determina cuántos píxeles en la imagen binarizada corresponden a las manchas negras.
3) Calcula el porcentaje de área de manchas negras en la imagen total dividiendo el número de píxeles de manchas por el tamaño total de la imagen binarizada y multiplicando por 100.
4) Utiliza **cv2.findContours** para identificar y contar las manchas negras. Esta función busca los contornos de las manchas negras en la imagen invertida. Los contornos representan las formas de las manchas y se almacenan en contornos_negros.
5) Calcula la cantidad de manchas negras contando el número de contornos encontrados en contornos_negros.
6) Prepara una cadena de texto Cadena que incluye la cantidad de manchas negras encontradas y el porcentaje de área cubierta por estas manchas.
7) Configura el cuadro de texto CajaTextoNegro en modo editable (state='normal'), borra su contenido anterior, inserta la cadena Cadena y vuelve a deshabilitar la edición del cuadro de texto (state='disabled').
8) La función **contar_manchas_blancas()** es similar a **contar_manchas_negras**() pero se enfoca en el conteo y el porcentaje de manchas blancas en lugar de negras. Ambas funciones siguen el mismo principio, pero difieren en el valor umbral y la lógica para encontrar las manchas en la imagen binarizada.

En resumen La función **contar_manchas_negras()** analiza una imagen umbralizada en busca de manchas negras. Primero, invierte la imagen, calcula el número de píxeles con manchas negras, determina el porcentaje de área ocupada por estas manchas y encuentra los

contornos de las manchas negras en la imagen. Luego, muestra la cantidad de manchas y el porcentaje de área en un cuadro de texto.

Por otro lado, la función **contar_manchas_blancas**() realiza un proceso similar, pero se centra en el conteo y el porcentaje de manchas blancas en la imagen binarizada. Ambas funciones siguen un proceso básico de análisis de manchas, diferenciándose únicamente en el umbral y la lógica de detección de manchas.

```
def contar_manchas_negras():
    # Invertir la imagen para contar manchas negras
    imagen_invertida = cv2.bitwise_not(bin_image)

    # Contar el número de píxeles con manchas negras
    num_pixels_con_manchas_negras = cv2.countNonZero(imagen_invertida)
    # Calcular el porcentaje de manchas
    porcentaje_manchas = (num_pixels_con_manchas_negras / bin_image.size) * 100

    # Contar manchas negras
    contornos_negros, _ = cv2.findContours(imagen_invertida, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    cantidad_manchas_negras = len(contornos_negros)
    Cadena = f"Cantidad de manchas negras: {cantidad_manchas_negras}\nPorcentaje área con manchas: {round(porcentaje_manchas, 2)}%"
    CajaTextoNegro.configure(state='normal')
    CajaTextoNegro.delete(1.0, tk.END)
    CajaTextoNegro.insert(1.0, Cadena)
    CajaTextoNegro.configure(state='disabled')

# Función para contar manchas blancas en la imagen
def contar_manchas_blancas():
    # Contar el número de píxeles con manchas blancas
    num_pixels_con_manchas_blancas = cv2.countNonZero(bin_image)
    # Calcular el porcentaje de manchas
    porcentaje_manchas = (num_pixels_con_manchas_blancas / bin_image.size) * 100

    # Contar manchas blancas
    contornos_blancos, _ = cv2.findContours(bin_image, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    cantidad_manchas_blancas = len(contornos_blancos)

    Cadena = f"Cantidad de manchas blancas: {cantidad_manchas_blancas}\nPorcentaje área con manchas: {round(porcentaje_manchas, 2)}%"
    CajaTextoBlanco.configure(state='normal')
    CajaTextoBlanco.delete(1.0, tk.END)
    CajaTextoBlanco.insert(1.0, Cadena)
    CajaTextoBlanco.configure(state='disabled')
```

En resumen, el programa proporciona una herramienta simple para realizar análisis de manchas en imágenes en escala de grises, lo que puede ser útil en diversas aplicaciones, como identificación de objetos o detección de áreas específicas en imágenes.

Al compilar y ejecutar el programa se verá así:

Y al analizar una imagen se verá así: