UNIVERSIDAD DE GUADALAJARA

CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERIAS

ALGORITMO 3

VISION ROBOTICA

INGENIERIA EN ROBOTICA

Alumno: Carlos Omar Rodriguez Vazquez

Introducción

El presente reporte documenta el desarrollo y los resultados obtenidos en el marco del Proyecto Integrador correspondiente a la Unidad 2. El objetivo principal de este proyecto consistió en la implementación de un programa capaz de identificar y clasificar el tipo de exposición de una imagen, ya sea normal, subexpuesta o sobreexpuesta, mediante el análisis de su histograma. Además, se propuso la aplicación de técnicas de ecualización de imágenes para mejorar la calidad visual en casos de subexposición o sobreexposición. Este proyecto constituye una aplicación práctica de los conceptos y habilidades adquiridos durante el estudio de la unidad, contribuyendo así al desarrollo de competencias en el procesamiento de imágenes digitales.

Desarrollo

Exposición de la imagen

Con el propósito de optimizar el código, se ha optado por desarrollar un par de funciones para este programa, siendo una de ellas la encargada de determinar la exposición de una imagen, ya sea subexpuesta, sobreexpuesta o con exposición normal.

```
def exp_imagen(image):
ig = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
[M,N] = ig.shape[0:2]

hist = cv.calcHist([ig], [0], None, [5], [0,256]).flatten()/(M*N)

maxElement = np.argmax(hist)
if maxElement == 4 and hist[4] > 0.3:
exp = 'subexpuesta'
    elif maxElement == 0 and hist[0] > 0.3:
exp = 'sobreexpuesta'
else:
        exp = 0

return [ig, exp, hist]
```

Esta función toma como parámetro la imagen original, la cual se somete a un proceso de conversión a escala de grises, permitiendo así obtener las dimensiones de la imagen. Posteriormente, se calcula el histograma dividido en quintiles y se normaliza. La siguiente fase del algoritmo se encarga de determinar la exposición de la imagen, identificando el elemento más frecuente en el histograma. Si el elemento más frecuente corresponde al contenedor 4 y supera un umbral predefinido de 0.3, se clasifica la imagen como subexpuesta. De manera análoga, si el elemento más frecuente es el contenedor 0 y supera el mismo umbral, se clasifica como sobreexpuesta. En caso contrario, se considera que la imagen posee una exposición normal. Finalmente, la función retorna la imagen en escala de grises, la clasificación de exposición y el histograma en quintiles, facilitando así futuros procesamientos dentro del algoritmo.

Ecualización de la imagen

Una función adicional implementada en este código se encarga de llevar a cabo la ecualización de la imagen.

```
def equa_image(ig):
[M,N] = ig.shape[0:2]
hist = cv.calcHist([ig], [0], None, [256], [0,256]).flatten()/(M*N)
    ImE = np.zeros_like(ig)

    FA = np.zeros_like(hist)

    for i in range(len(FA)):
        FA[i] = FA[i-1] + hist[i]

    for i in range(M):
        for j in range(N):
            ImE[i,j] = 255*FA[ig[i,j]]

    return [ImE, hist]
```

Esta función recibe como parámetro la imagen en escala de grises y obtiene sus dimensiones. A continuación, calcula el histograma de la imagen con 256 divisiones y lo normaliza. Se inicializa una matriz para la imagen ecualizada con las mismas dimensiones que la imagen original, pero con todos sus valores establecidos en 0.

Posteriormente, se calcula la frecuencia acumulada para cada valor de intensidad de píxeles en el histograma, lo que facilita el proceso de ecualización. La ecualización de la imagen se realiza iterando sobre cada píxel de la imagen original y aplicando una regla definida por la expresión: ImE[píxel] = L * FA[ig[píxel]], donde L representa el número total de niveles de intensidad (en este caso, 256) y FA[ig[píxel]] es el valor de la frecuencia acumulada en la posición correspondiente al valor de intensidad del píxel en la imagen original.

Finalmente, la función devuelve la imagen ecualizada y el histograma asociado a esta imagen, proporcionando así los resultados necesarios para futuros procesos dentro del algoritmo.

Agregar texto a la imagen: Función

Se ha implementado una función adicional en este algoritmo para agregar texto a una imagen. Aunque ya existe una función específica en OpenCV para esta tarea, se consideró importante desarrollar una función personalizada debido a la frecuente utilización de esta operación en el código, con parámetros bastante similares.

```
def agregar_text(image, text, color, escala, grosor):
    [M, N] = image.shape[0:2]
    print(M, N)
    margen = 20
    font = cv.FONT_HERSHEY_COMPLEX

    size_text, _ = cv.getTextSize(text, font, escala, grosor)

    x_pos = N - size_text[0] - margen
    y_pos = M - margen

    cv.putText(image, text, (x_pos, y_pos), font, escala, color, grosor)
```

Esta función recibe como parámetros los mismos que la función de OpenCV, con la diferencia de la posición. Esta discrepancia es uno de los principales motivos para la creación de esta función personalizada, ya que aquí se calcula la posición del texto con base en el tamaño de la imagen y del texto, asegurando así que se escriba en la esquina inferior derecha de la imagen.

Inicio del programa y configuración inicial

El programa se inicia con las operaciones básicas necesarias, que incluyen la importación de las librerías requeridas, la lectura de la imagen a procesar y la inicialización de algunas variables esenciales.

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

n_image = 3
image = f'im{n_image}.png'

Im = cv.imread(image)
ImE = None
negro = (0,0,0)
blanco = (255,255,255)
```

Las librerías empleadas en este código son cv2, numpy y matplotlib, cada una desempeñando un papel específico en el procesamiento de imágenes. Para facilitar la optimización y la reutilización del código con distintas imágenes, se ha establecido un formato estándar de nomenclatura para los archivos de imagen. Todos los archivos de imagen siguen el mismo patrón de nomenclatura: 'im[número de imagen].png'. Antes de leer la imagen, se genera dinámicamente el nombre del archivo concatenando el valor de la variable n_image. Esto permite cambiar fácilmente la imagen de entrada modificando simplemente el valor de n_image. Posteriormente, la imagen se lee utilizando la función cv.imread() y se asigna a la variable Im. Además, se inicializa la variable ImE con el valor None, la cual adquirirá significado en etapas posteriores del código.

Exposición de la imagen y agregar texto a la imagen: Implementación

En esta sección se encuentra el núcleo del algoritmo, donde se llevan a cabo todos los procesos y se emplean las funciones desarrolladas.

```
[ig, exp, hist] = exp_imagen(Im)

if exp == 'subexpuesta':
    [ImE, hist2] = equa_image(ig)
    agregar_text(ig, 'Imagen subexpuesta', blanco, 1, 2)
    agregar_text(ImE, 'Imagen equalizada por OR', blanco, 1, 2)
    cv.imshow('imagen equalizada', ImE)
elif exp == 'sobreexpuesta':
    [ImE, hist2] = equa_image(ig)
    agregar_text(ig, 'Imagen sobreexpuesta', negro, 1, 2)
    agregar_text(ImE, 'Imagen equalizada por OR', negro, 1, 2)
    cv.imshow('imagen equalizada', ImE)
else:
    agregar_text(ig, 'Imagen con exposicion normal', negro, 1, 2)
```

El código comienza utilizando la función exp_imagen() para obtener la imagen en escala de grises, la exposición de la imagen y su histograma correspondiente, los cuales serán utilizados para ecualizar la imagen y agregar texto. Se emplean un par de condicionales para realizar ciertos procesos: si la imagen está subexpuesta o sobreexpuesta, se lleva a cabo un proceso de ecualización con la función equa_image(), ya que no es necesario ecualizar una imagen con exposición normal. Posteriormente, dependiendo del tipo de exposición de la imagen, se escribe en la imagen en escala de grises el tipo de exposición correspondiente. Finalmente, si la imagen fue ecualizada, se añade el texto 'imagen equalizada por OR' en la imagen ecualizada.

Resultados

Imagen 1

Figura 1: Foto de AURORA en la de grabacion para la cancion 'Storm' utilizada como la Imagen 1

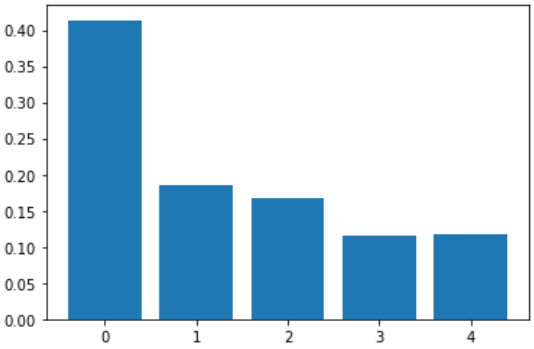

Figura 2: Histograma de la Imagen 1

Como era de esperar al observar la imagen, el histograma exhibe una alta frecuencia en los colores cercanos a 0, lo que indica que la imagen está subexpuesta.

(a) (b)

Figura 3: (a) Imagen 1 en escala de grises con el tipo de exposicion (b) Imagen equalizada.

Imagen 2

Figura 4: Imagen de Moana como la Imagen 2

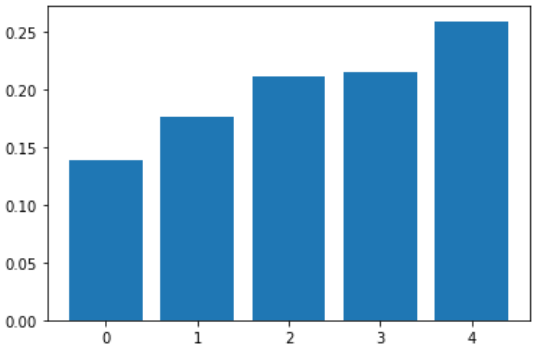

Figura 5: Histograma de la Imagen 2

Como era de esperar al observar la imagen, el histograma exhibe una frecuencia distribuida, lo que indica que la imagen tiene una exposicion normal.

Figura 6: Imagen 2 en escala de grises con el tipo de exposicion

Imagen 3

Figura 7: Portada del comic 'Dark Crisis on Infinite Earths' como Imagen 3

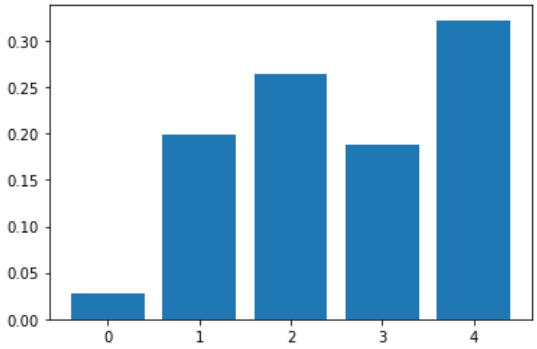

Figura 8: Histograma de la Imagen 3

Como era de esperar al observar la imagen, el histograma exhibe una alta frecuencia en los colores cercanos a 255, lo que indica que la imagen está sobreexpuesta.

(a) (b)

Figura 9: (a) Imagen 1 en escala de grises con el tipo de exposicion (b) Imagen equalizada.

Conclusión

La implementación del Proyecto Integrador correspondiente a la Unidad 2 ha concluido con éxito, proporcionando resultados significativos en el análisis y procesamiento de imágenes digitales. A través de la aplicación de técnicas avanzadas, se logró desarrollar un programa capaz de identificar y clasificar el tipo de exposición de una imagen, así como mejorar su calidad visual mediante la ecualización. Este proyecto ha permitido consolidar los conocimientos adquiridos durante el estudio de la unidad, demostrando la aplicación práctica de conceptos fundamentales en el campo de la visión artificial y la robótica. Además, se destaca la importancia de la optimización del código y la implementación de funciones personalizadas para aumentar la eficiencia y la flexibilidad del algoritmo. En resumen, este proyecto representa un paso significativo hacia el dominio de técnicas avanzadas de procesamiento de imágenes y sienta las bases para futuras investigaciones y aplicaciones en este emocionante campo de estudio.