Nombre: Edgar Javier Fregoso Cuarenta

Registro: 22310285

Practica: Histograma y ecualizado del histograma

Materia : visión Artificial

1. Objetivo de la práctica

Aplicar la técnica de **ecualización del histograma** a imágenes digitales utilizando la librería OpenCV en Python, con el fin de mejorar el contraste visual de las imágenes obtenidas en la Práctica 2.

2. Fundamento teórico

La **ecualización de histograma** es una técnica del procesamiento digital de imágenes que permite mejorar el contraste de una imagen, redistribuyendo los niveles de intensidad de los píxeles. En imágenes en escala de grises, esta técnica puede aplicarse directamente. Sin embargo, en imágenes a color (RGB), no se recomienda aplicar la ecualización directamente a cada canal por separado, ya que puede generar distorsiones de color.

Una alternativa más adecuada es convertir la imagen al espacio de color **YUV**, aplicar la ecualización únicamente al canal **Y** (luminancia), y luego reconvertir la imagen al formato original. OpenCV proporciona herramientas como cvtColor() para el cambio de espacio de color, y equalizeHist() para realizar la ecualización.

3. Desarrollo de la práctica

Se utilizó Python junto con OpenCV para procesar tres imágenes diferentes. A continuación, se presenta el código utilizado:

```
import cv2

import numpy

img1 = cv2.imread('3D-Matplotlib.png')

img_to_yuv = cv2.cvtColor(img1, cv2.COLOR_BGR2YUV)

img_to_yuv[:,:,0] = cv2.equalizeHist(img_to_yuv[:,:,0])

hist_equalization_result1 = cv2.cvtColor(img_to_yuv, cv2.COLOR_YUV2BGR)

cv2.imwrite('result.jpg', hist_equalization_result1)

img2 = cv2.imread('mainlogo.png')

img_to_yuv = cv2.cvtColor(img2, cv2.COLOR_BGR2YUV)
```

```
img_to_yuv[:,:,0] = cv2.equalizeHist(img_to_yuv[:,:,0])
hist_equalization_result2 = cv2.cvtColor(img_to_yuv, cv2.COLOR_YUV2BGR)
cv2.imwrite('result1.jpg', hist_equalization_result2)
img3 = cv2.imread('mainsvmimage.png')
img_to_yuv = cv2.cvtColor(img3, cv2.COLOR_BGR2YUV)
img_to_yuv[:,:,0] = cv2.equalizeHist(img_to_yuv[:,:,0])
hist_equalization_result3 = cv2.cvtColor(img_to_yuv, cv2.COLOR_YUV2BGR)
cv2.imwrite('result2.jpg', hist_equalization_result3)
cv2.imshow('Imagen 1', hist_equalization_result1)
cv2.imshow('Imagen 2', hist_equalization_result2)
cv2.imshow('Imagen 3', hist_equalization_result3)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

4. Resultados

Después de aplicar la ecualización de histograma, se obtuvo una mejora notable en el contraste de las tres imágenes. Las áreas oscuras y claras se distribuyeron de manera más uniforme, lo que permite una mejor visualización de los detalles.

Se generaron y guardaron tres imágenes resultantes:

- result.jpg (correspondiente a la imagen 3D-Matplotlib)
- result1.jpg (correspondiente a mainlogo)
- result2.jpg (correspondiente a mainsvmimage)

5. Comentarios y análisis

OpenCV simplifica la implementación de técnicas de procesamiento de imágenes gracias a funciones como equalizeHist(). Sin embargo, dado que las imágenes a color poseen tres canales (BGR), aplicar ecualización directamente sobre cada uno puede

alterar el balance cromático. Por ello, se convierte la imagen al espacio YUV para modificar solo la luminancia (canal Y), preservando así la integridad del color.

Este enfoque fue mencionado en el libro *Python: Real World Machine Learning*, y demostró ser efectivo y sencillo. Las funciones utilizadas fueron:

- cv2.cvtColor() para convertir el espacio de color.
- cv2.equalizeHist() para aplicar la ecualización.
- cv2.imwrite() y cv2.imshow() para guardar y visualizar resultados.

---

6. Conclusiones

- La ecualización de histograma mejora el contraste de imágenes de manera significativa.
- En imágenes a color, es mejor aplicar esta técnica solo al canal de luminancia (Y) en el espacio YUV.
- OpenCV proporciona funciones directas y eficientes para realizar este tipo de operaciones, lo cual es fundamental en aplicaciones de procesamiento de imágenes, visión artificial y aprendizaje automático.