CENTRO DE ENSEÑANZA TÉCNICA INDUSTRIAL

INGENIERÍA EN MECATRÓNICA

Visión artificial

Práctica 3:
Histograma y ecualizado del histograma

Alumna:
Vanessa Aguirre Diaz

Registro:
22310274

Fecha:
21 de abril del 2025

Desarrollo teórico:

¿Qué es un histograma?

El histograma es una representación gráfica de los distintos tonos que contiene una imagen y su distribución, en el que sobre el eje de abscisas se representan los diferentes valores que pueden tomar los píxeles de una imagen y en el eje de ordenadas el número de píxeles que se encuentran en la imagen para ese valor de cuantización se conoce como histograma de los niveles de cuantización de la imagen, o simplemente histograma de la imagen.

Se debe notar que los histogramas no dicen nada sobre la disposición espacial de los píxeles. Por ello, un histograma es una forma de representación de imágenes en la que se produce pérdida de información. A partir del histograma de una imagen es imposible deducir la imagen que lo originó. Se deriva que aunque una imagen sólo puede tener un histograma, imágenes diferentes podrían tener el mismo histograma.

El histograma de una imagen en niveles de gris proporciona información sobre el número de píxeles que hay para cada nivel de intensidad (ver Figura 35). En imágenes en color RGB se usan 3 histogramas, uno por cada componente de color.

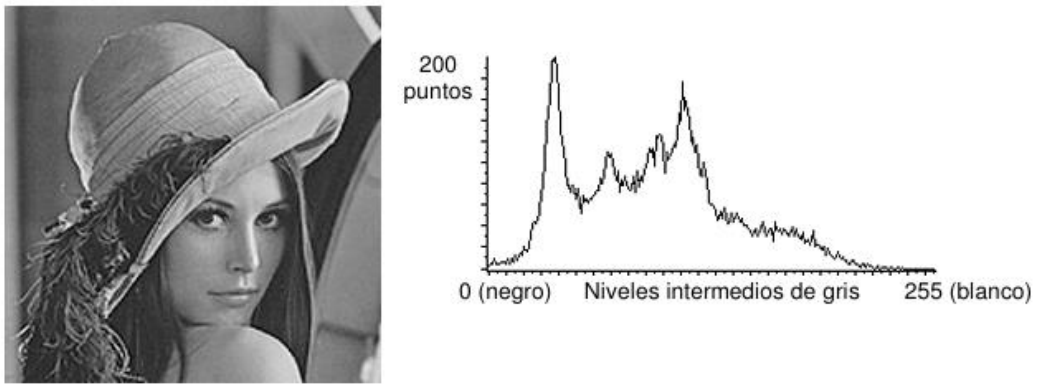


Figura 35.- Imagen en niveles de gris de Lena y su correspondiente histograma.

2.9.2 Ecualizado del histograma.

La ecualización de histograma es una técnica que ajusta los valores de los píxeles de una imagen según su histograma de intensidad, lo que resulta en una imagen con una distribución uniforme de intensidades y un histograma plano. Esta técnica mejora la visibilidad de los detalles de la imagen al aprovechar todo el rango dinámico.

La ecualización del histograma es una técnica de procesamiento de imágenes que mejora el contraste ajustando la distribución de los niveles de intensidad (brillo) de una imagen. Su objetivo principal es redistribuir los valores de píxeles para que se aproveche de forma más

uniforme todo el rango de intensidades posibles (de 0 a 255 en imágenes de 8 bits). Esto hace que las zonas oscuras se aclaren y las zonas claras se oscurezcan ligeramente, revelando detalles que antes no eran visibles, especialmente en imágenes con poco contraste o iluminación desigual.

Características:

- Aumenta el contraste en imágenes con intensidades concentradas en una región estrecha del histograma.
- No necesita parámetros: se adapta automáticamente a la imagen.
- Es una técnica global: afecta a todos los píxeles por igual.
- Funciona mejor en imágenes en escala de grises.
- Puede producir efectos no deseados en imágenes a color si no se aplica cuidadosamente.

Efectos sobre la imagen:

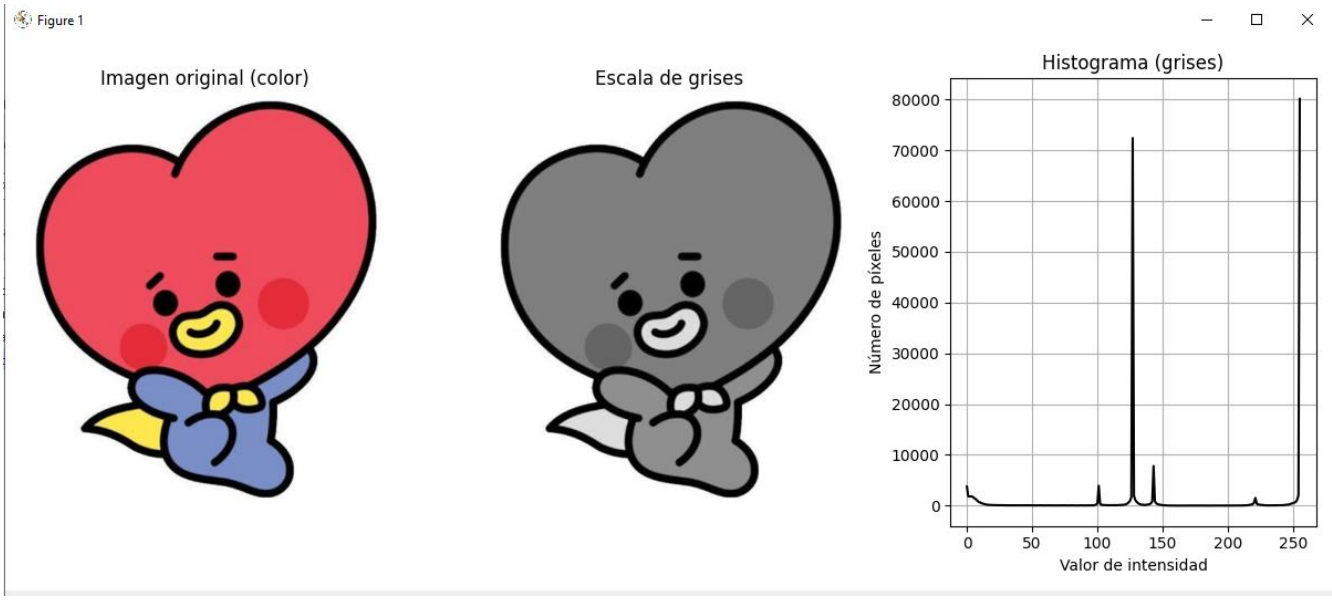
- Mejora visualmente los detalles ocultos.
- Realza bordes y texturas.
- Puede hacer que la imagen luzca más nítida o más "dramática".
- En imágenes a color, si se aplica por canal individual, puede cambiar los colores reales y hacerlos irreales.

Desarrollo práctico

Primero tenemos el código en Python para obtener el histograma de una imagen en blanco y negro:

```
# Practica 3: Histograma de una imagen

# Importar librerías necesarias
import cv2  # OpenCV para procesamiento de imágenes
import matplotlib.pyplot as plt  # Matplotlib para mostrar imágenes y graficar

# Cargar imagen a color
img1 = cv2.imread('tata.jpg')  # Cambia 'tata.jpg' por la ruta de tu imagen

# Convertir a escala de grises
img2 = cv2.imread('tata.jpg', cv2.IMREAD_GRAYSCALE)

# Convertir imagen BGR a RGB (para que se vea bien en matplotlib)
img1_rgb = cv2.cvtColor(img1, cv2.COLOR_BGR2RGB)
```

```
# Calcular el histograma de la imagen en escala de grises
hist = cv2.calcHist([img2], [0], None, [256], [0, 256])

# Crear una figura para mostrar las imágenes y el histograma
plt.figure(figsize=(12, 5))  # Ancho x Alto en pulgadas

# Mostrar imagen original (a color, convertida a RGB)
plt.subplot(1, 3, 1)  # 1 fila, 3 columnas, posición 1
plt.imshow(img1_rgb)
plt.title('Imagen original (color)')
plt.axis('off')  # Oculta los ejes

# Mostrar imagen en escala de grises
plt.subplot(1, 3, 2)  # Posición 2
plt.imshow(img2, cmap='gray')
plt.title('Escala de grises')
plt.axis('off')

# Mostrar histograma de la imagen en escala de grises
plt.subplot(1, 3, 3)  # Posición 3
plt.plot(hist, color='black')
plt.title('Histograma (grises)')
plt.xlabel('Valor de intensidad')
plt.ylabel('Número de píxeles')
plt.grid(True)

# Ajustar el diseño para que no se encimen los elementos
plt.tight_layout()

# Mostrar todo
plt.show()


# Esperar a que el usuario presione una tecla
cv2.waitKey(0)

# Cerrar todas las ventanas abiertas de OpenCV
cv2.destroyAllWindows()
```

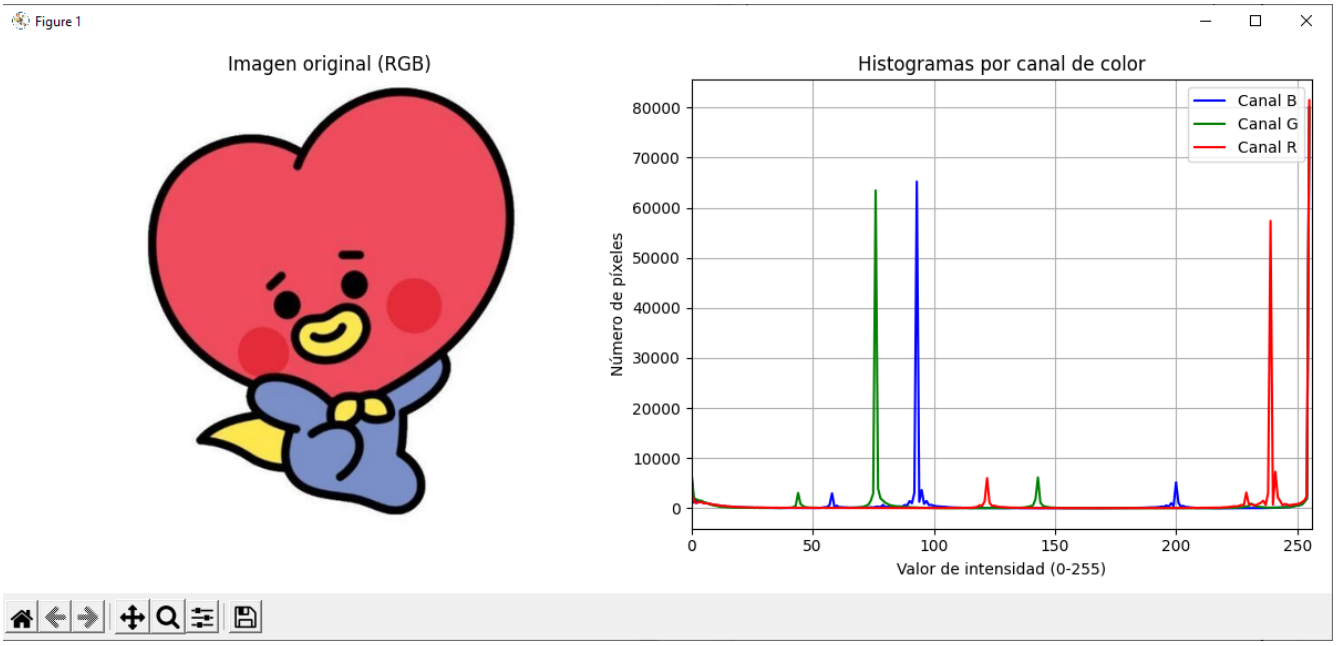


Para la segunda parte, se ajusta el código para obtener el histograma de cada uno de los canales de color RGB de la imagen:

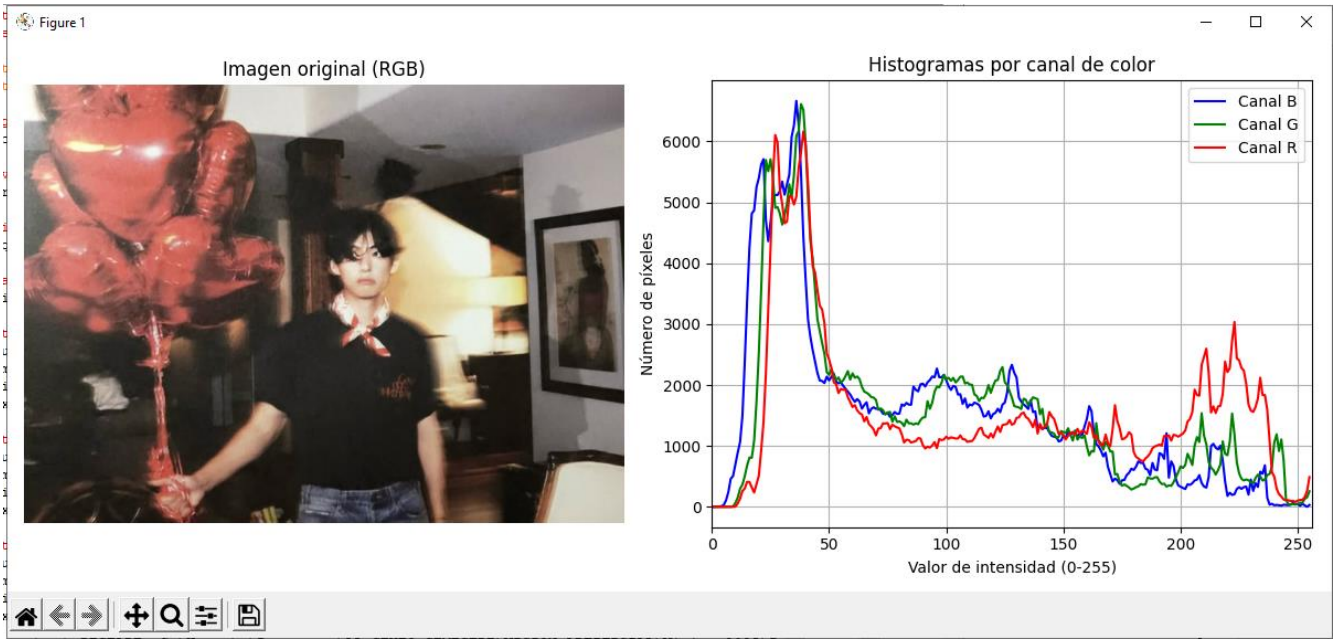
```
# Practica 3: Histograma de una imagen

# Importar librerías
import cv2  # OpenCV para procesamiento de imágenes
import matplotlib.pyplot as plt  # Matplotlib para mostrar imágenes y graficar histogramas

# Cargar la imagen en color (OpenCV la carga en formato BGR por defecto)
imagen = cv2.imread('tata.jpg')

# Convertir la imagen de BGR a RGB (para que se vea correctamente con matplotlib)
imagen_rgb = cv2.cvtColor(imagen, cv2.COLOR_BGR2RGB)

# Definir los canales de color: azul (b), verde (g), rojo (r)
canales = ('b', 'g', 'r')  # BGR: Blue, Green, Red en OpenCV

# Diccionario que relaciona cada canal con su color para graficar
colores = {'b': 'blue', 'g': 'green', 'r': 'red'}

# Crear una figura para mostrar la imagen y los histogramas
plt.figure(figsize=(12, 5))  # Tamaño de la figura (ancho x alto en pulgadas)

# Mostrar la imagen original en color (convertida a RGB)
plt.subplot(1, 2, 1)  # 1 fila, 2 columnas, posición 1
plt.imshow(imagen_rgb)
plt.title('Imagen original (RGB)')
plt.axis('off')  # Ocultar ejes

# Crear el segundo subplot para los histogramas
```

```
plt.subplot(1, 2, 2)  # Posición 2

# Iterar sobre cada canal y graficar su histograma
for i, canal in enumerate(canales):
    # Calcular el histograma del canal actual
    hist = cv2.calcHist([imagen], [i], None, [256], [0, 256])

    # Dibujar el histograma con el color correspondiente
    plt.plot(hist, color=colores[canal], label=f'Canal {canal.upper()}')

# Configurar gráfico
plt.title('Histogramas por canal de color')
plt.xlabel('Valor de intensidad (0-255)')
plt.ylabel('Número de píxeles')
plt.xlim([0, 256])  # Límite del eje X
plt.legend()  # Mostrar leyenda
plt.grid(True)  # Mostrar cuadrícula de fondo

# Ajustar el diseño y mostrar todo
plt.tight_layout()
plt.show()
```

Resultado con una caricatura de colores solidos:



Resultado con una fotografía:

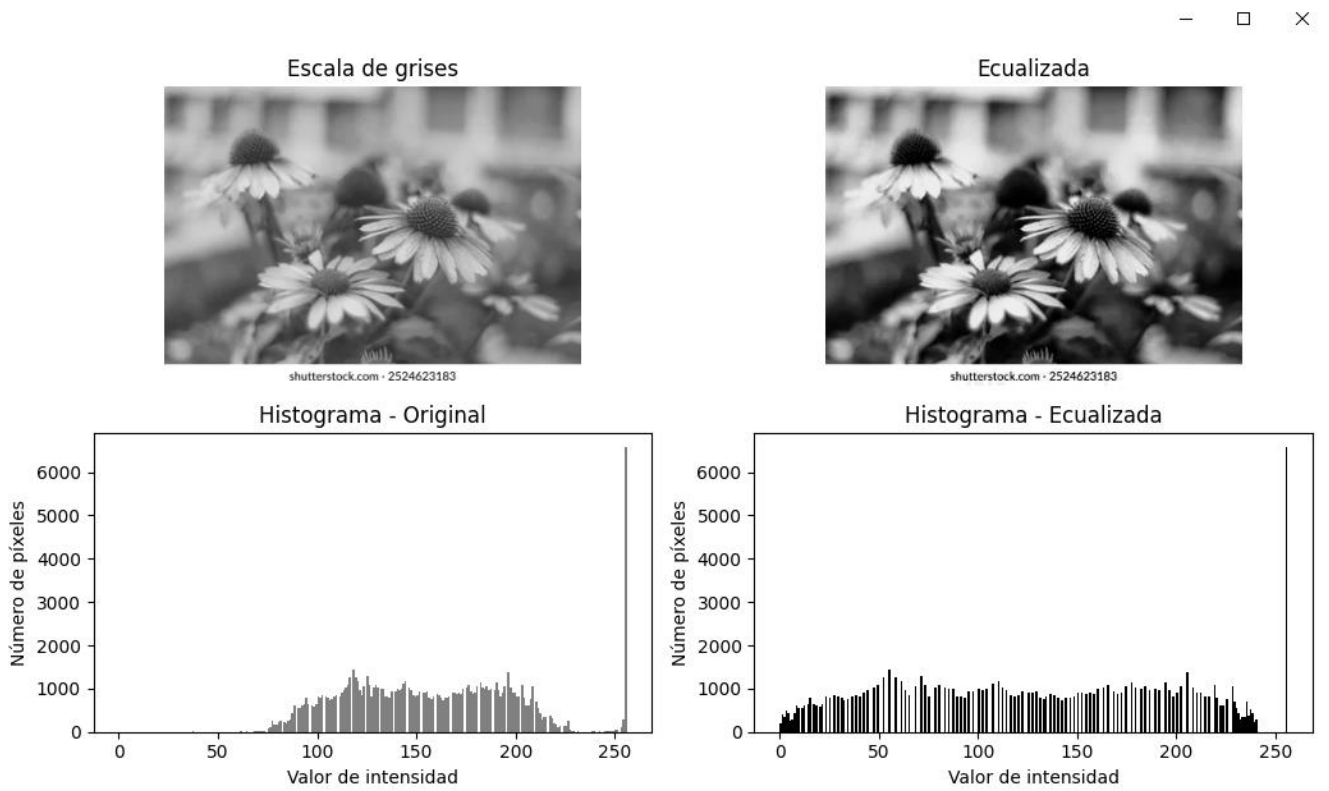


Ecualización del histograma:

La parte 3 de la practica consiste en ecualizar el histograma de una imagen y ver que efecto tiene la ecualización sobre esta:

```
#Practica 3 Pt.2: Ecualizacion del histograma
# La ecualización de histograma solo puede AUMENTAR el contraste, nunca reducirlo.

import cv2  # OpenCV para procesamiento de imágenes
import matplotlib.pyplot as plt  # Matplotlib para mostrar imágenes y gráficos

# Cargar imagen en color
img_color = cv2.imread('flores.jpg')  # Cambia el nombre por el de tu imagen

# Convertir a escala de grises
img_gray = cv2.cvtColor(img_color, cv2.COLOR_BGR2GRAY)

# Aplicar ecualización de histograma
img_eq = cv2.equalizeHist(img_gray)

# Crear una figura para mostrar las imágenes
plt.figure(figsize=(15, 6))

# Mostrar imagen original en color (convertida de BGR a RGB para que se vea bien con matplotlib)
plt.subplot(2, 3, 1)
```

```
plt.imshow(cv2.cvtColor(img_color, cv2.COLOR_BGR2RGB))
plt.title('Original (Color)')
plt.axis('off')  # Quitar ejes

# Mostrar imagen en escala de grises
plt.subplot(2, 3, 2)
plt.imshow(img_gray, cmap='gray')
plt.title('Escala de grises')
plt.axis('off')

# Mostrar imagen ecualizada
plt.subplot(2, 3, 3)
plt.imshow(img_eq, cmap='gray')
plt.title('Ecualizada')
plt.axis('off')

# Histograma de la imagen en escala de grises original
plt.subplot(2, 3, 5)
plt.hist(img_gray.ravel(), 256, [0, 256], color='gray')
plt.title('Histograma - Original')
plt.xlabel('Valor de intensidad')
plt.ylabel('Número de píxeles')

# Histograma de la imagen ecualizada
plt.subplot(2, 3, 6)
plt.hist(img_eq.ravel(), 256, [0, 256], color='black')
plt.title('Histograma - Ecualizada')
plt.xlabel('Valor de intensidad')
plt.ylabel('Número de píxeles')

# Ajustar distribución de los subplots
plt.tight_layout()
plt.show()

# Esperar a que el usuario presione una tecla
cv2.waitKey(0)

# Cerrar todas las ventanas abiertas
cv2.destroyAllWindows()
```



Como podemos ver, la ecualización aumenta el contraste de la imagen haciendo que pequeños detalles que antes no se veían o se perdían en la imagen ahora si puedan distinguirse más claramente.