ceti

CENTRO DE ENSEÑANZA
TÉCNICA INDUSTRIAL

Maestro: Mauricio Alejandro Cabrera Arellano

Alumno: Alejandro Retana Rubio 22110315

Materia: Visión Artificial

Practica 8

Fecha: 09-06-2025

Práctica: Detección de Bordes en Imágenes

En esta práctica se implementaron distintos métodos de detección de bordes sobre una imagen en escala de grises. El objetivo fue comparar cómo cada técnica resalta los contornos y transiciones en la imagen:

- **Laplaciano**: Detecta bordes en todas direcciones usando la segunda derivada. Es sensible a cambios bruscos en la intensidad de píxeles.
- **Sobel X y Sobel Y**: Detectan bordes en direcciones específicas. Sobel X resalta bordes verticales, mientras que Sobel Y identifica bordes horizontales.
- **Canny**: Es un método más avanzado que incluye suavizado, detección de gradientes y umbralización para identificar bordes delgados y bien definidos.

Estas técnicas son fundamentales en visión por computadora, ya que permiten segmentar objetos, analizar formas y preparar imágenes para tareas posteriores como reconocimiento o clasificación.

Codigo

```
# Importar librerías necesarias
import cv2  # OpenCV para procesamiento de imágenes
import numpy as np  # Operaciones numéricas
import matplotlib.pyplot as plt  # Visualización de imágenes

# Cargar la imagen en escala de grises
img = cv2.imread('luffy5.png', cv2.IMREAD_GRAYSCALE)
# Se carga la imagen en escala de grises para facilitar la detección de bordes

# --------- Método 1: Laplaciano ---------
laplaciano = cv2.Laplacian(img, cv2.CV_64F)
# Detecta bordes en todas direcciones aplicando la segunda derivada
laplaciano = cv2.convertScaleAbs(laplaciano)
# Convierte a valores absolutos en escala de 8 bits para visualización

# --------- Método 2: Sobel X ---------
sobelx = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=3)
# Detecta bordes verticales (cambios horizontales en la imagen)
sobelx = cv2.convertScaleAbs(sobelx)
# Convierte a escala de 8 bits

# --------- Método 3: Sobel Y ---------
sobely = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize=3)
# Detecta bordes horizontales (cambios verticales en la imagen)
sobely = cv2.convertScaleAbs(sobely)
# Convierte a escala de 8 bits

# --------- Método 4: Canny ---------
canny = cv2.Canny(img, 100, 200)
# Aplica el algoritmo de Canny con umbrales de histéresis 100 y 200

# --------- Visualización de resultados ---------
plt.figure(figsize=(10, 6))  # Configura el tamaño del gráfico

# Imagen original
plt.subplot(2, 3, 1)
plt.imshow(img, cmap='gray')
plt.title('Imagen Original')
plt.axis('off')
```

```
        # Resultado Laplaciano
        plt.subplot(2, 3, 2)
plt.imshow(laplaciano, cmap='gray')
      plt.title('Laplaciano')
          plt.axis('off')

        # Resultado Sobel X
        plt.subplot(2, 3, 3)
  plt.imshow(sobelx, cmap='gray')
      plt.title('Sobel X')
         plt.axis('off')

        # Resultado Sobel Y
        plt.subplot(2, 3, 4)
  plt.imshow(sobely, cmap='gray')
      plt.title('Sobel Y')
         plt.axis('off')

         # Filtro Canny
        plt.subplot(2, 3, 5)
 plt.imshow(canny, cmap='gray')
       plt.title('Canny')
         plt.axis('off')

# Ajustamos el diseño para que no se encimen las imágenes
                plt.tight_layout()
                    plt.show()
```

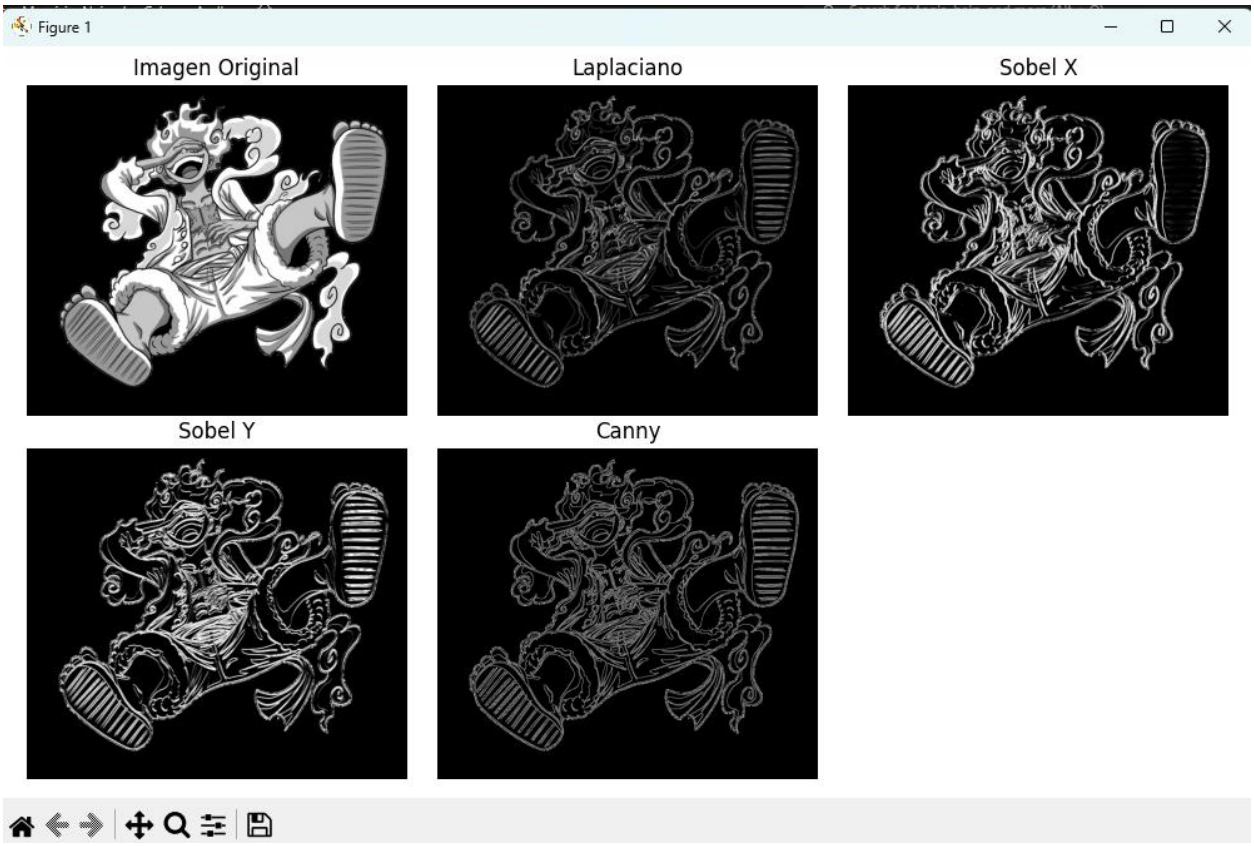
Figure 1
Imagen Original
Laplaciano
Sobel X
Sobel Y
Canny