Reporte de Práctica: Detección de Bordes en Video en Tiempo Real con OpenCV

Nombre del estudiante: Leonardo
Carrera: Ingeniería Mecatrónica
Especialidad: Biomédica
Materia: Visión Artificial
Fecha: 27 de mayo de 2025
Nombre de la práctica: Detección de Bordes en Tiempo Real con Filtros Laplaciano, Sobel y Canny

---

Objetivo

Implementar un sistema de visión artificial que procese video en tiempo real utilizando la biblioteca OpenCV, aplicando filtros clásicos de detección de bordes (Laplaciano, Sobel y Canny) para comparar su funcionamiento y observar los resultados visualmente.

Material y Herramientas

- Lenguaje de programación: Python 3.x
- Biblioteca: OpenCV (cv2), NumPy (numpy)
- Hardware: Laptop con cámara web integrada
- IDE recomendado: Visual Studio Code / Jupyter Notebook / PyCharm

---

Fundamento Teórico

La detección de bordes es una operación fundamental en visión artificial para extraer información estructural de una imagen. Algunos de los operadores más comunes son:

- **Filtro Laplaciano**: Calcula la segunda derivada, útil para encontrar regiones donde cambia rápidamente la intensidad.
- **Filtro Sobel**: Calcula la primera derivada en las direcciones X e Y. Es útil para identificar bordes horizontales y verticales.

- **Detector Canny**: Algoritmo más robusto que combina suavizado, gradientes, supresión de no-máximos y umbral con histéresis para obtener bordes definidos y continuos.

Procedimiento

1. Se importaron las bibliotecas necesarias (cv2, numpy).
2. Se capturó video en tiempo real desde la cámara web.
3. Se convirtió cada fotograma a escala de grises.
4. Se aplicaron los siguientes detectores de bordes:
    - Laplaciano
    - Sobel X
    - Sobel Y
    - Canny
5. Se mostraron los resultados en ventanas separadas para visualización comparativa.
6. El programa se cerró al presionar la tecla **'q'**.

Código Implementado

python

CopiarEditar

```
import cv2
import numpy as np


cap = cv2.VideoCapture(0)


while True:
  ret, frame = cap.read()
```

```
    if not ret:
        break

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    laplacian = cv2.Laplacian(gray, cv2.CV_64F)
    laplacian = cv2.convertScaleAbs(laplacian)

    sobelx = cv2.Sobel(gray, cv2.CV_64F, 1, 0, ksize=3)
    sobely = cv2.Sobel(gray, cv2.CV_64F, 0, 1, ksize=3)

    sobelx = cv2.convertScaleAbs(sobelx)
    sobely = cv2.convertScaleAbs(sobely)

    canny = cv2.Canny(gray, 100, 200)

    cv2.imshow('Original', frame)
    cv2.imshow('Laplaciano', laplacian)
    cv2.imshow('Sobel X', sobelx)
    cv2.imshow('Sobel Y', sobely)
    cv2.imshow('Canny', canny)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
```

```
cv2.destroyAllWindows()
```

Resultados

Se observaron cinco ventanas:

- Imagen original
- Imagen con detección de bordes usando Laplaciano
- Sobel en X (bordes verticales)
- Sobel en Y (bordes horizontales)
- Canny (bordes más definidos y precisos)

Esto permitió comparar visualmente cómo cada operador responde ante los cambios de intensidad en una imagen en movimiento.

Conclusión

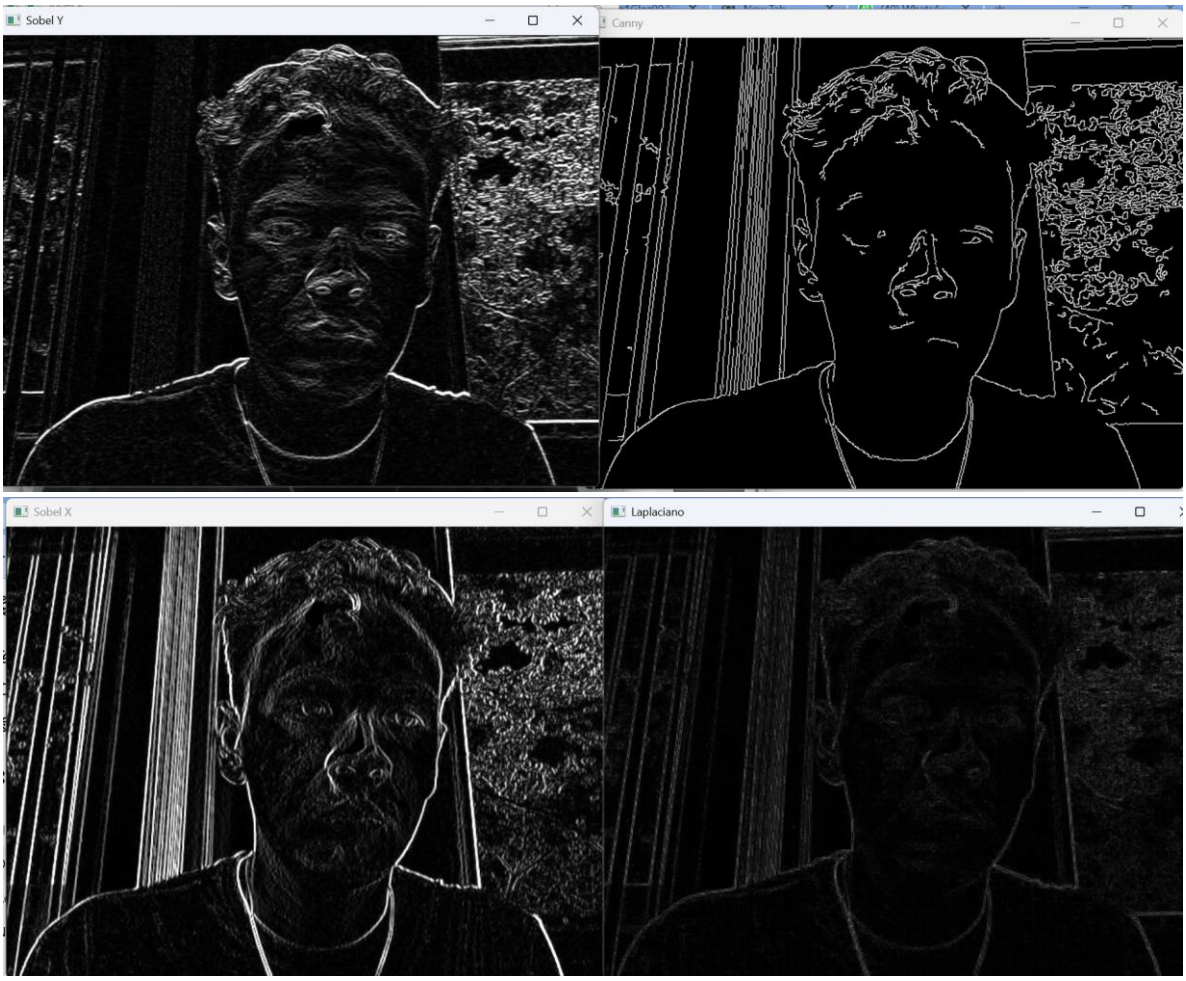
Los detectores de bordes son herramientas esenciales en la visión artificial. Cada uno tiene sus propias ventajas:

- **Sobel** es útil para detectar orientación específica de bordes.
- **Laplaciano** es más sensible pero puede generar más ruido.
- **Canny** proporciona los resultados más limpios y útiles para tareas posteriores como segmentación o reconocimiento.

Esta práctica permitió aplicar conceptos teóricos en tiempo real, facilitando su comprensión e identificación de casos de uso según el contexto.

Observaciones

- Se recomienda tener buena iluminación para mejores resultados.
- En condiciones de poca luz, el detector Canny muestra mejor desempeño.
- Puede extenderse la práctica para aplicar los bordes detectados a sistemas de reconocimiento de objetos o seguimiento.



Link repositorio:

https://github.com/Leo1Glez99/Visi-n-Artificial/tree/main/Practica_008