



Rodrigo Hernández Zavala

27 de mayo de 2025

22310215

Practica 8: Detección de bordes

1. Inicialización de la cámara

```
python
CopiarEditar
cap = cv2.VideoCapture(0)
```

- `cv2.VideoCapture(0)` abre la cámara predeterminada (la de tu laptop).
 - `cap.read()` se usa dentro del ciclo para capturar cada fotograma (imagen) del video.
-

2. Lectura y preprocesamiento del fotograma

```
python
CopiarEditar
frame = cv2.resize(frame, (640, 480))
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

- Se **redimensiona** la imagen para acelerar el procesamiento.
 - Luego se **convierte a escala de grises**, ya que la detección de bordes no necesita color, solo intensidad de luz.
-

3. Aplicación de detectores de bordes

◆ A. Laplaciano

```
python
CopiarEditar
laplacian = cv2.Laplacian(gray, cv2.CV_64F)
laplacian = cv2.convertScaleAbs(laplacian)
```

- Detecta todos los bordes sin importar la dirección.
 - Usa derivadas de segundo orden → sensible al ruido.
 - `cv2.convertScaleAbs` convierte los valores negativos a positivos para visualización.
-

◆ B. Sobel X

```
python
CopiarEditar
sobelx = cv2.Sobel(gray, cv2.CV_64F, 1, 0, ksize=5)
sobelx = cv2.convertScaleAbs(sobelx)
```

- Detecta bordes **verticales** (cambios de intensidad en eje X).
- 1, 0 indica que se calcula la derivada en X.

◆ C. Sobel Y

```
python
CopiarEditar
sobelY = cv2.Sobel(gray, cv2.CV_64F, 0, 1, ksize=5)
sobelY = cv2.convertScaleAbs(sobelY)
```

- Detecta bordes **horizontales** (cambios en el eje Y).
-

◆ D. Canny

```
python
CopiarEditar
canny = cv2.Canny(gray, 100, 200)
```

- Detector de bordes **más preciso y robusto**.
 - Usa:
 - **Filtro Gaussiano** → reduce ruido
 - **Gradientes** para bordes fuertes
 - **Supresión de no máximos**
 - **Umbráles** de histéresis (100 = bajo, 200 = alto)
 - Es el **más usado en visión artificial real**.
-



4. Visualización en ventanas separadas

```
python
CopiarEditar
cv2.imshow("Original", frame)
cv2.imshow("Laplaciano", laplacian)
cv2.imshow("Sobel X", sobelX)
cv2.imshow("Sobel Y", sobelY)
cv2.imshow("Canny", canny)
```

Cada filtro se muestra **en una ventana distinta**, así puedes:

- Comparar los resultados en tiempo real.
 - Analizar qué método capta mejor los bordes deseados según la escena.
-



5. Salir con una tecla

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

- El programa se detiene cuando presionas la tecla **q**.
-

¿Qué puedes concluir?

| Filtro | Ventaja principal | Limitación principal |
|-------------------|--------------------------|----------------------------------|
| Laplaciano | Simple y rápido | Muy sensible al ruido |
| Sobel X/Y | Buen control direccional | Puede dejar pasar bordes débiles |
| Canny | Más completo y preciso | Más costoso computacionalmente |

Código:

```
import cv2

import numpy as np


# Inicializar la cámara

cap = cv2.VideoCapture(0)

if not cap.isOpened():

    print("Error: No se pudo acceder a la cámara.")

    exit()

while True:

    ret, frame = cap.read()

    if not ret:

        break

    # Redimensionar para mejor rendimiento
```

```
frame = cv2.resize(frame, (640, 480))

# Convertir a escala de grises
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

# ----- Detección de Bordes -----

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

# Sobel X
sobelx = cv2.Sobel(gray, cv2.CV_64F, 1, 0, ksize=5)
sobelx = cv2.convertScaleAbs(sobelx)

# Sobel Y
sobely = cv2.Sobel(gray, cv2.CV_64F, 0, 1, ksize=5)
sobely = cv2.convertScaleAbs(sobely)

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

# ----- Mostrar resultados -----

cv2.imshow("Original", frame)
cv2.imshow("Laplaciano", laplacian)
```

```

cv2.imshow("Sobel X", sobelx)

cv2.imshow("Sobel Y", sobely)

cv2.imshow("Canny", canny)

# Salir con 'q'

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

    break

cap.release()

cv2.destroyAllWindows()

```

Imagenes del funcionamiento:

