Universidad Autónoma de San Luis Potosí
Facultad de ingeniería
Tratamiento de Imágenes
Practica 5
Nombre Práctica: Filtrado en el Dominio Espacial
Nombre del Alumno: Manuel Ramírez Galván

Fecha: 25/02/2025

Procedimiento

3.1. Diseñe un programa en Python en el cual se cumplan los siguientes requisitos:

- Obtenga la captura de video de una cámara web conectada a la computadora (interna o externa).
- La captura de video debe de ser mostrado en la ventana creada con el nombre "Ejercicio 5.1" hasta que se presione la tecla q.
- Aplicar el filtro suavizante de caja sobre el video al presionar la tecla b.
- Aplicar el filtro suavizante Gaussiano sobre el video al presionar la tecla g.
- Aplicar el filtro suavizante medio sobre el video al presionar la tecla

Resultados

```python
import cv2

cam = cv2.VideoCapture(0)

apply_blur = False
apply_gauss = False
apply_med = False

while cam.isOpened():
    ret, frame = cam.read()
    if not ret:
        break

    if apply_blur:
        frame = cv2.blur(src=frame, ksize=(11, 11))

    if apply_gauss:
        frame = cv2.GaussianBlur(src=frame, ksize=(11, 11), sigmaX=0,
sigmaY=0)
    if apply_med:
        frame = cv2.medianBlur(frame, ksize=11)

    cv2.imshow("Ejercicio 5.1", frame)

    key = cv2.waitKey(1)

    if key == ord('q'):
        break
    elif key == ord('n'):
        apply_blur = False
        apply_gauss = False
        apply_med = False
    elif key == ord('b'):
        apply_blur = True
        apply_gauss = False
        apply_med = False
    elif key == ord('g'):
        apply_gauss = True
        apply_blur = False
        apply_med = False
    elif key == ord('m'):
        apply_med = True
        apply_blur = False
        apply_gauss = False

cam.release()
cv2.destroyAllWindows()
```

Imagen 1.- Código Ejercicio 1

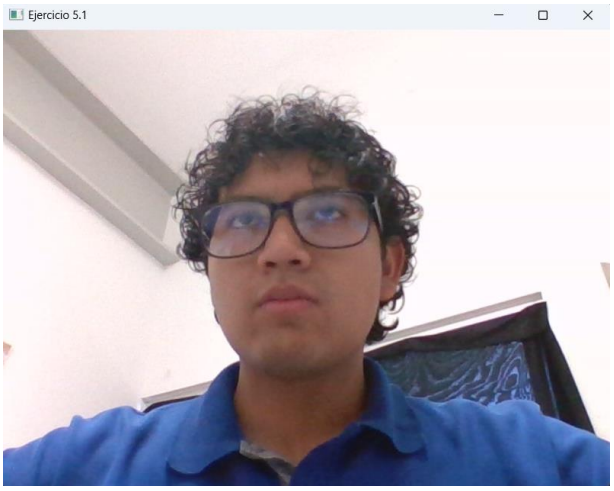

Imagen 2.- Cámara sin Filtro

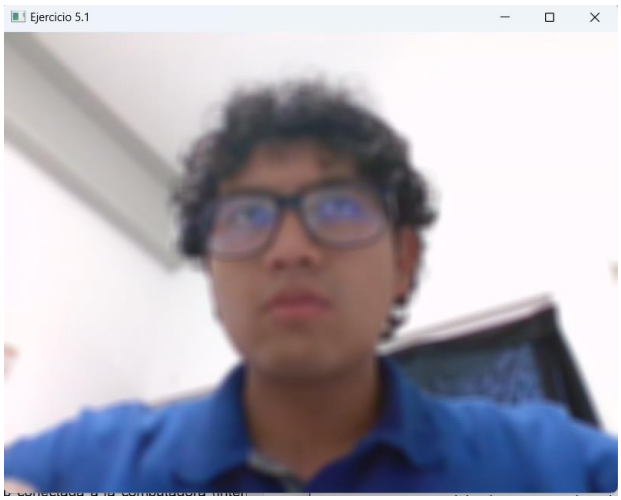

Imagen 3.- Cámara con Filtro Simple

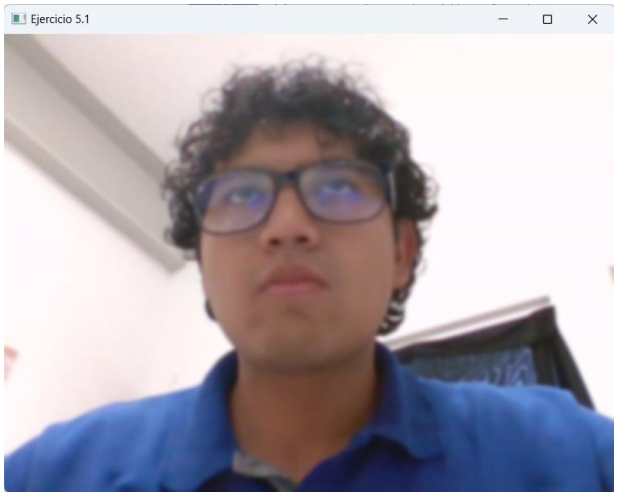

Imagen 4.- Cámara con Filtro Gaussiano

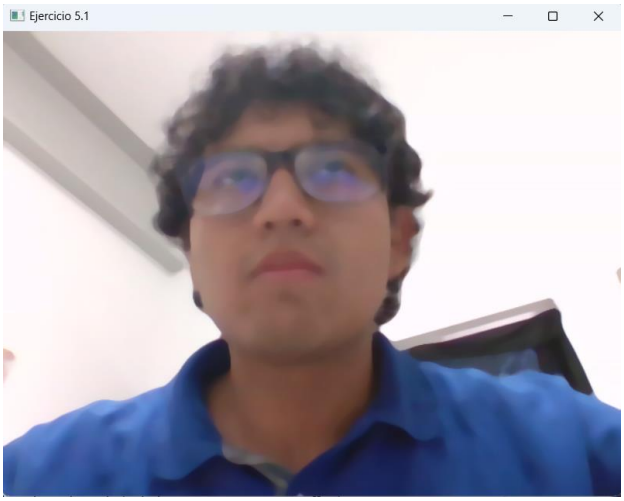

Imagen 5.- Cámara con Filtro Medio

Procedimiento

3.2. Diseñe un programa en Python en el cual se cumplan los siguientes requisitos:

- Obtenga la captura de video de una cámara web conectada a la computadora (interna o externa).
- La captura de video debe de ser mostrado en la ventana creada con el nombre "Ejercicio 5.2" hasta que se presione la tecla q.
- Aplicar el filtro de detección de bordes Sobel el video al presionar la tecla s.
- Aplicar el filtro de detección de bordes Prewitt sobre el video al presionar la tecla p.
- Aplicar el de detección de bordes Roberts sobre el video al presionar la tecla r.
- Aplicar el de detección de bordes Canny sobre el video al presionar la tecla c.
- Al presionar la tecla n, regresar a la captura de video sin filtros.

Resultados

```python
import cv2
import numpy as np

cam = cv2.VideoCapture(0)

apply_sobel = False
apply_prewitt = False
apply_roberts = False
apply_canny = False

while cam.isOpened():
    ret, frame = cam.read()
    if not ret:
        break

    if apply_sobel:
        frame = cv2.Sobel(src= frame, ddepth=-1, dx=1, dy=1, ksize=5)

    if apply_prewitt:
        prew_kernel_x = np.array([[-1, 0, 1],
                                  [-1, 0, 1],
                                  [-1, 0, 1]])

        prew_kernel_y = np.array([[-1, -1, -1],
                                  [0, 0, 0],
                                  [1, 1, 1]])

        der_x = cv2.filter2D(frame, cv2.CV_64F, prew_kernel_x)
        der_y = cv2.filter2D(frame, cv2.CV_64F, prew_kernel_y)

        absX = cv2.convertScaleAbs(der_x)
        absY = cv2.convertScaleAbs(der_y)

        frame = cv2.addWeighted(absX, 0.5, absY, 0.5, 0)

    if apply_roberts:
        rob_kernel_x = np.array([[1, 0],
                                 [0, -1]])

        rob_kernel_y = np.array([[0, 1],
                                 [-1, 0]])

        der_x = cv2.filter2D(frame, cv2.CV_64F, rob_kernel_x)
        der_y = cv2.filter2D(frame, cv2.CV_64F, rob_kernel_y)

        absX = cv2.convertScaleAbs(der_x)
        absY = cv2.convertScaleAbs(der_y)

        frame = cv2.addWeighted(absX, 0.5, absY, 0.5, 0)

    if apply_canny:
        frame = cv2.Canny(image=frame, threshold1=100,
threshold2=200)
    cv2.imshow("Ejercicio 5.2", frame)

    key = cv2.waitKey(1)

    if key == ord('q'):
        break
    elif key == ord('n'):
        apply_sobel = False
        apply_prewitt = False
        apply_roberts = False
        apply_canny = False
    elif key == ord('s'):
        apply_sobel = True
        apply_prewitt = False
        apply_roberts = False
        apply_canny = False
    elif key == ord('p'):
        apply_prewitt = True
        apply_sobel = False
        apply_roberts = False
        apply_canny = False
    elif key == ord('r'):
        apply_roberts = True
        apply_prewitt = False
        apply_sobel = False
        apply_canny = False
    elif key == ord('c'):
        apply_canny = True
        apply_sobel = False
        apply_roberts = False
        apply_prewitt = False

cam.release()
cv2.destroyAllWindows()
```

Imagen 6.- Código Ejercicio 2

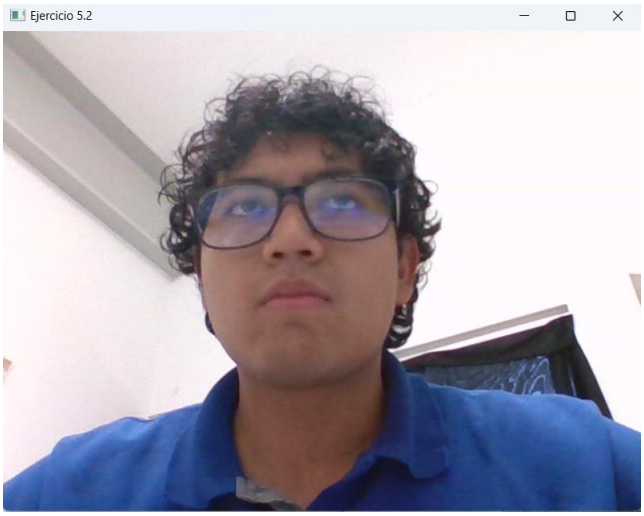

Imagen 7.- Cámara sin Filtro

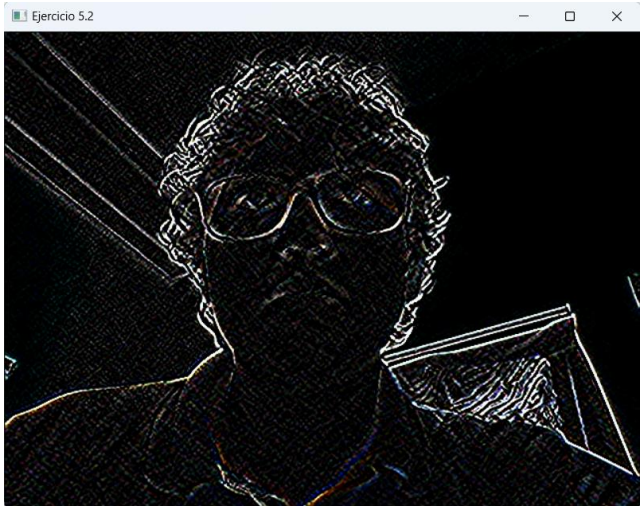

Imagen 7.- Cámara con Filtro Sobel

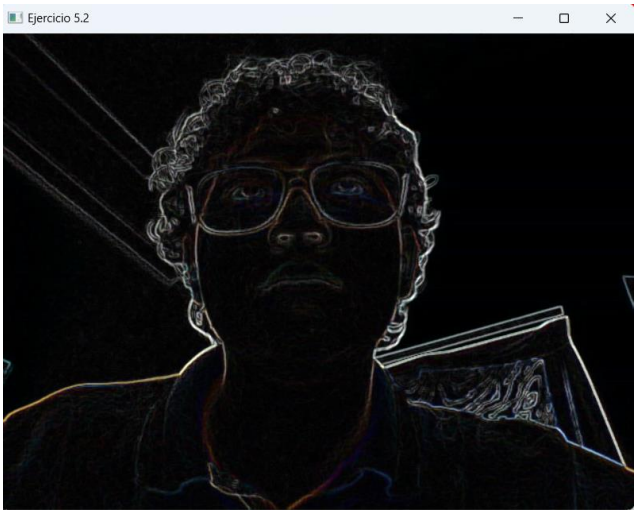

Imagen 7.- Cámara con Filtro Prewitt

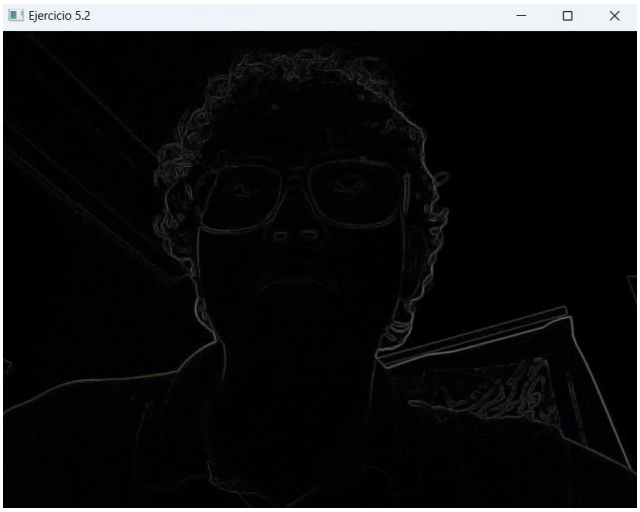

Imagen 7.- Cámara con Filtro Roberts

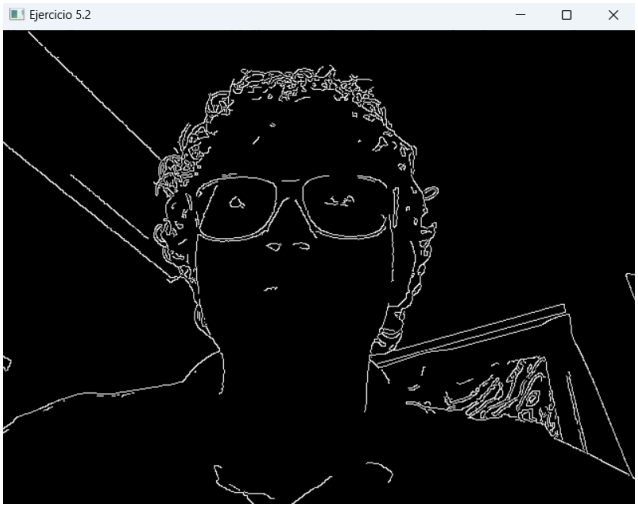

Imagen 7.- Cámara con Filtro Canny

Procedimiento

5.3. Modifique los valores de tamaño de kernel de convolución y parametros de entrada. Observe y describa las diferencias.

Resultados

Al aumentar el kernel en el primer ejercicio, cada filtro esta mas desenfocado, hay mayor suavizado y al disminuirlo es menor el suavizado del filtro.

Al aumentarlo en el segundo, los bordes de cada filtro eran más delgados, llegando a no verse unos o que solo se veían al acercarse uno a la cámara. En el filtro de Sobel se vveia mas ruido al aumentarle el kernel.

Comprensión

1. **¿Cuáles son los filtros lineales y para que se utilizan?**

Son técnicas de procesamiento de imágenes que aplican una operación matemática lineal para, esto al tomar un pixel y calcular su nuevo valor con una combinación lineal de los pixeles vecinos.

Se utilizan para suavizado y eliminación de ruido, detección de bordes, mejora de contraste.

2. **¿Diferencia entre filtro Gaussiano y mediana?**

El filtro Gaussiano es un filtro linear y el de mediana, no; el filtro Gaussiano es bueno para el ruido gaussiano y el de mediana para el ruido Sal y Pimienta; El filtro Gaussiano difumina ligeramente los bordes y el de mediana conserva mejor los bordes; el gaussiano es más rápido en imágenes grandes y el de mediana más lento.

3. **¿Cuáles son los tipos de filtros de detección de bordes?**

Sobel: Detección de bordes en direcciones especificas

Prewitt: Rápido y simple

Roberts: Rápido y para bordes diagonales

Laplaciano: Detección de bordes en todas direcciones

Canny: Preciso y además elimina el ruido

Conclusiones

Existen filtros lineales como no lineales, como es el cado del Gaussiano y la Mediana, la aplicación de estos depende del tipo de ruido que exista en la imagen o en este caso en el video, además se puede aumentar el kernel para tener mayor suavizado.

Además existen filtros para la detección de bordes, en los cuales el que me gusto más como se veía es el de Prewitt, ya que el de Roberts casi no se distinguen todos los bordes y el de Sobel hay ruido, aunque el que pienso que es el mejor es el de Canny ya que se distinguen aún más los bordes y no hay tanto ruido, ya que hace un filtrado.