UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO

UNIDAD ACADÉMICA PROFESIONAL TIANGUISTENCO

e

Licenciatura en Ingeniería en Software

UNIDAD DE APRENDIZAJE:

TÉCNICAS Y MÉTODOS DE PROCESAMIENTO DE IMÁGENES

PROFESOR:

ROCIO ELIZABETH PULIDO ALBA

GRUPO:

S8

ALUMNO:

SAAVEDRA CABALLERO ROBERTO DANIEL

PERIODO ESCOLAR:

2023A

FECHA:

23/02/2023

ASIGNACIÓN:

PRIMER PARCIAL

Instrucciones:

De forma individual en Python realiza la activación de cámara, captura video en tiempo real procesa empleando opencv numpy y otras librerías.

Cada punto en ventana diferente, siempre en tiempo real muestra lo siguiente:

0 se visualiza la imagen normal

1 invierte la vista de la imagen tipo espejo (--><--)

2 la vista de la imagen generala tipo caricatura

3 la vista de la imagen se visualiza en canal GREEN (asegúrate que se vea verde)

4 la vista de la imagen se visualiza la adición de la imagen en caricatura con la imagen verde

5 la vista de la imagen original se le agrega una unbralizacion

6 la vista de la imagen en canal GREEN se le agrega una máscara

7 la vista de la imagen en tipo espejo se pinta de RED los fragmentos captados de ese color (asegúrese de que se vea RED)

8 la vista original agregar filtro sepia

9 la vista original agregar segmentación grabcut

10 de cada vista en la misma ventana muestra el histograma

Documenta tu proceso realiza Reporte de Practica

Realiza video mostrando los resultados y explicando código que desarrollaste

Sube tu evidencia

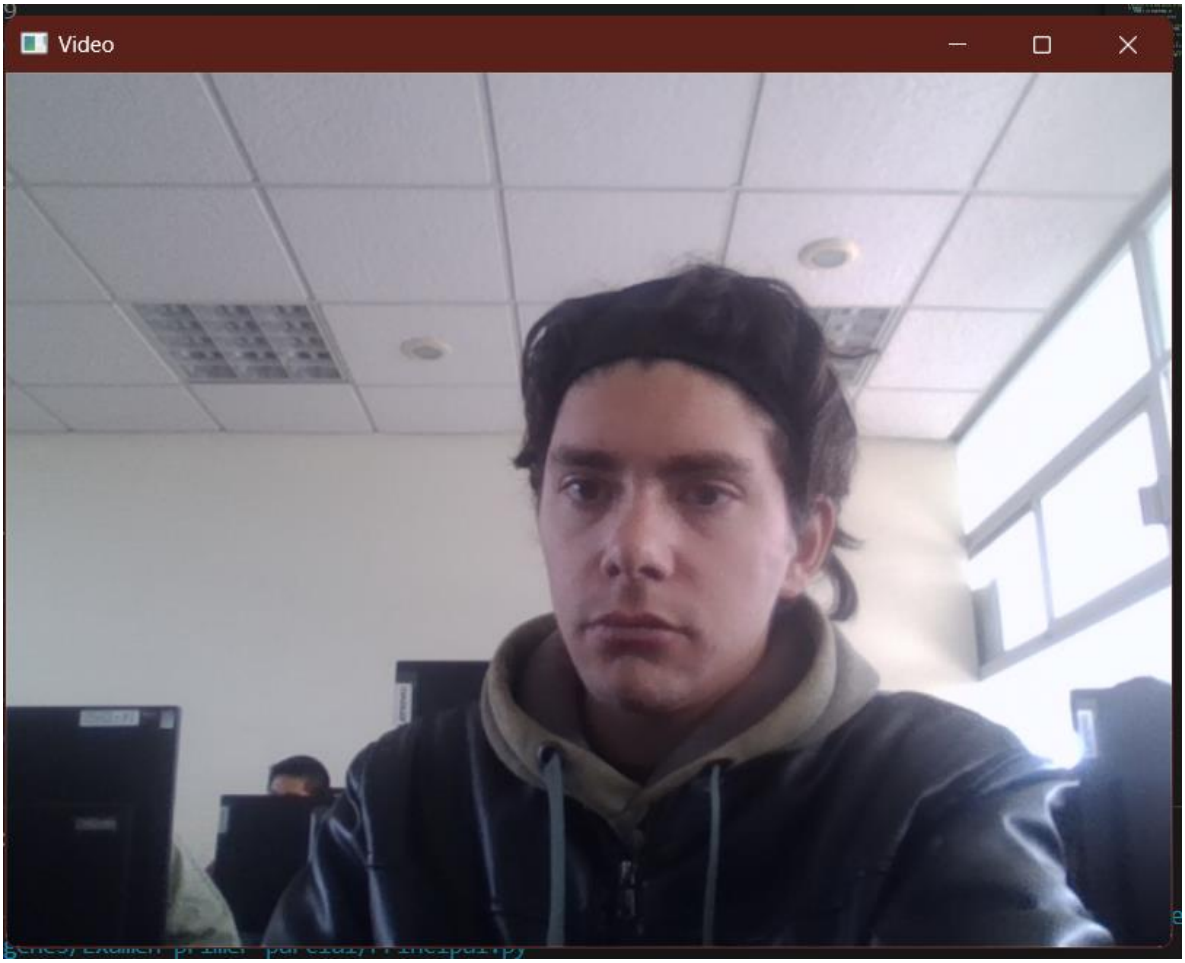
Video, código funcional, y Reporte

Nota: Toda entrega en desface resta 20 puntos es calificación final

0.- Visualización normal de imagen y captura de video:

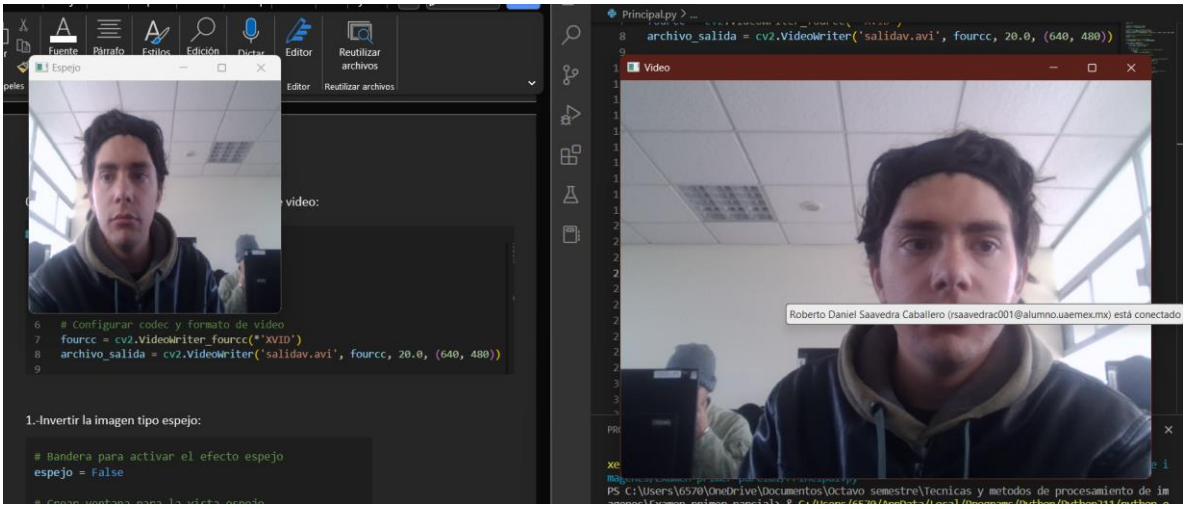
Principal.py > ...

```python
import cv2

# Capturar video de la cámara
captura = cv2.VideoCapture(0)

# Configurar codec y formato de video
fourcc = cv2.VideoWriter_fourcc(*'XVID')
archivo_salida = cv2.VideoWriter('salidav.avi', fourcc, 20.0, (640, 480))

```



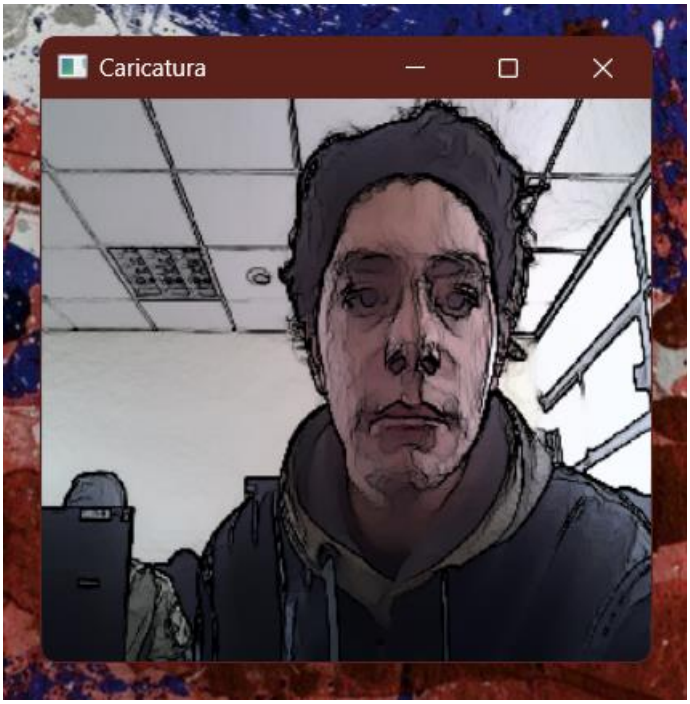
1.-Invertir la imagen tipo espejo:

```python
# Bandera para activar el efecto espejo
espejo = False

# Crear ventana para la vista espejo
cv2.namedWindow('Espejo', cv2.WINDOW_NORMAL)
```



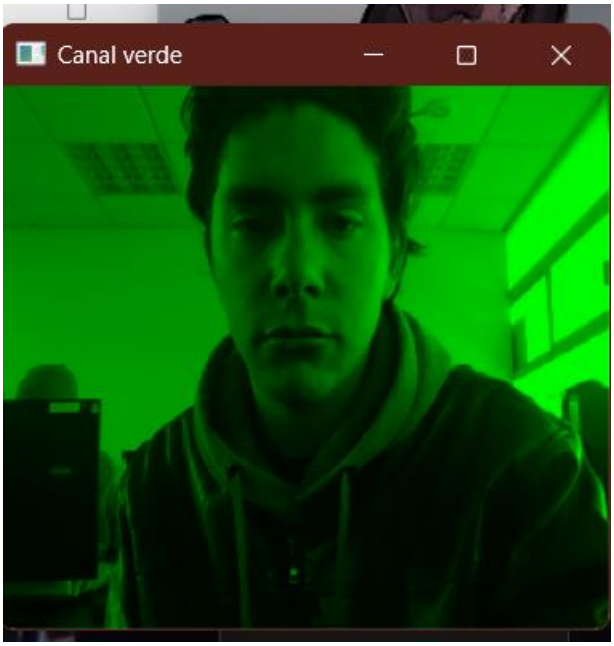
2.- Caricaturización de imagen:

```
# Aplicar efecto de caricatura
caricatura = cv2.stylization(frame, sigma_s=60, sigma_r=0.07)

# Mostrar el video en una ventana
cv2.imshow('Video',frame)

# Mostrar la vista espejo en la ventana correspondiente
cv2.imshow('Espejo', cv2.flip(frame, 1))

# Mostrar la vista de caricatura en la ventana correspondiente
cv2.imshow('Caricatura', caricatura)

# Guardar el video en archivo de salida
archivo_salida.write(frame)
```



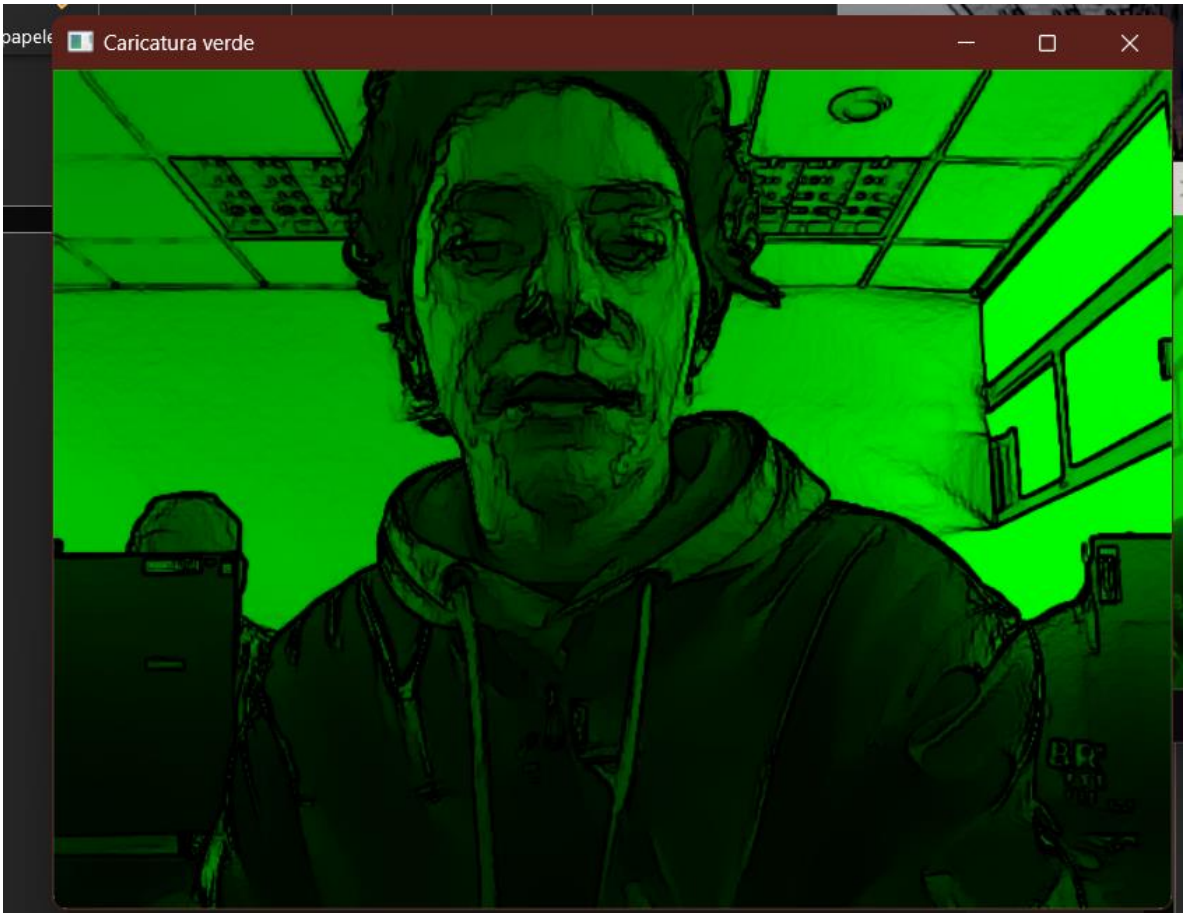
3.- Canal Green:

```
# Mostrar la vista espejo en la ventana correspondiente
cv2.imshow('Espejo', cv2.flip(frame, 1))

# Mostrar la vista de caricatura en la ventana correspondiente
cv2.imshow('Caricatura', caricatura)

# Mostrar la vista en canal GREEN en la ventana correspondiente
cv2.imshow('Canal verde', green)
```



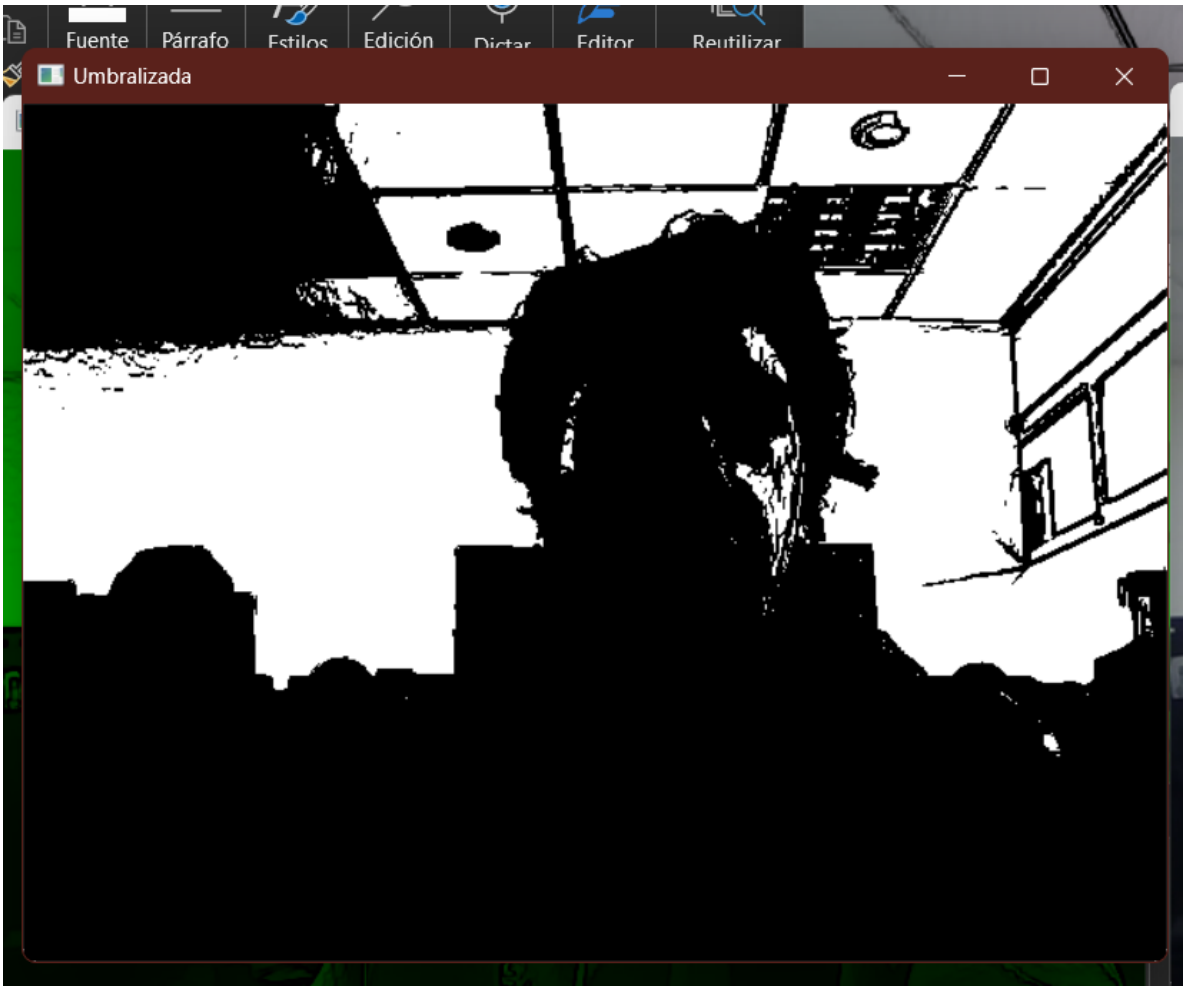
4.- Caricatura con green:

```python
# Obtener el canal verde y dejar los otros dos canales en cero
green = np.zeros(frame.shape, dtype=np.uint8)
green[:,:,1] = caricatura[:,:,1]

# Mostrar la vista en canal GREEN en la ventana correspondiente
cv2.imshow('Caricatura verde', green)

# Mostrar la vista de caricatura en la ventana correspondiente
cv2.imshow('Caricatura', caricatura)

# Obtener el canal verde y dejar los otros dos canales en cero
green = np.zeros(frame.shape, dtype=np.uint8)
green[:,:,1] = caricatura[:,:,1]

# Mostrar la vista en canal GREEN en la ventana correspondiente
cv2.imshow('Caricatura verde', green)
```

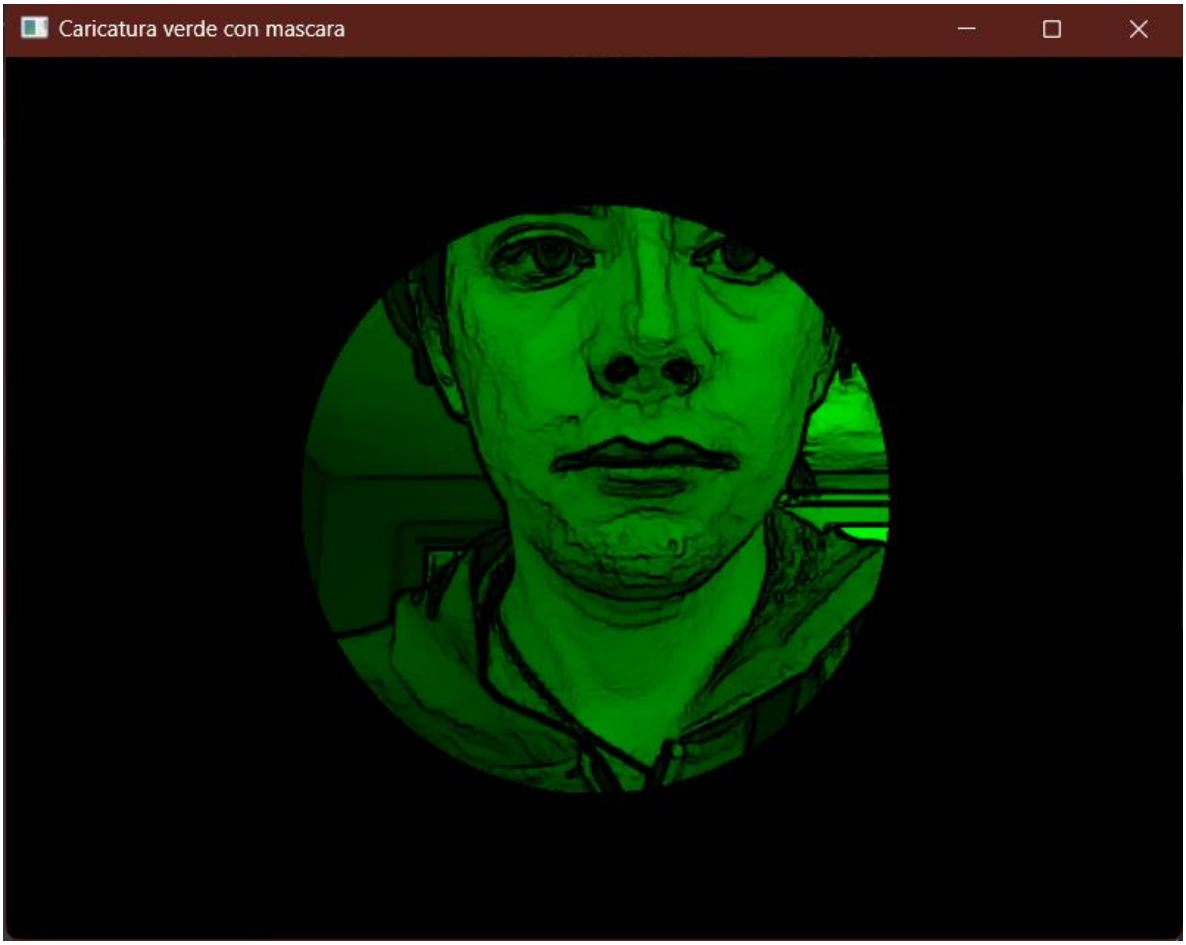


5.- Unbralización:

```
    # Aplicar umbral a la imagen caricatura
umbralizada = cv2.cvtColor(caricatura, cv2.COLOR_BGR2GRAY)
_, umbralizada = cv2.threshold(umbralizada, 127, 255, cv2.THRESH_BINARY)

# Mostrar la imagen umbralizada en la ventana correspondiente
cv2.imshow('Umbralizada', umbralizada)
```

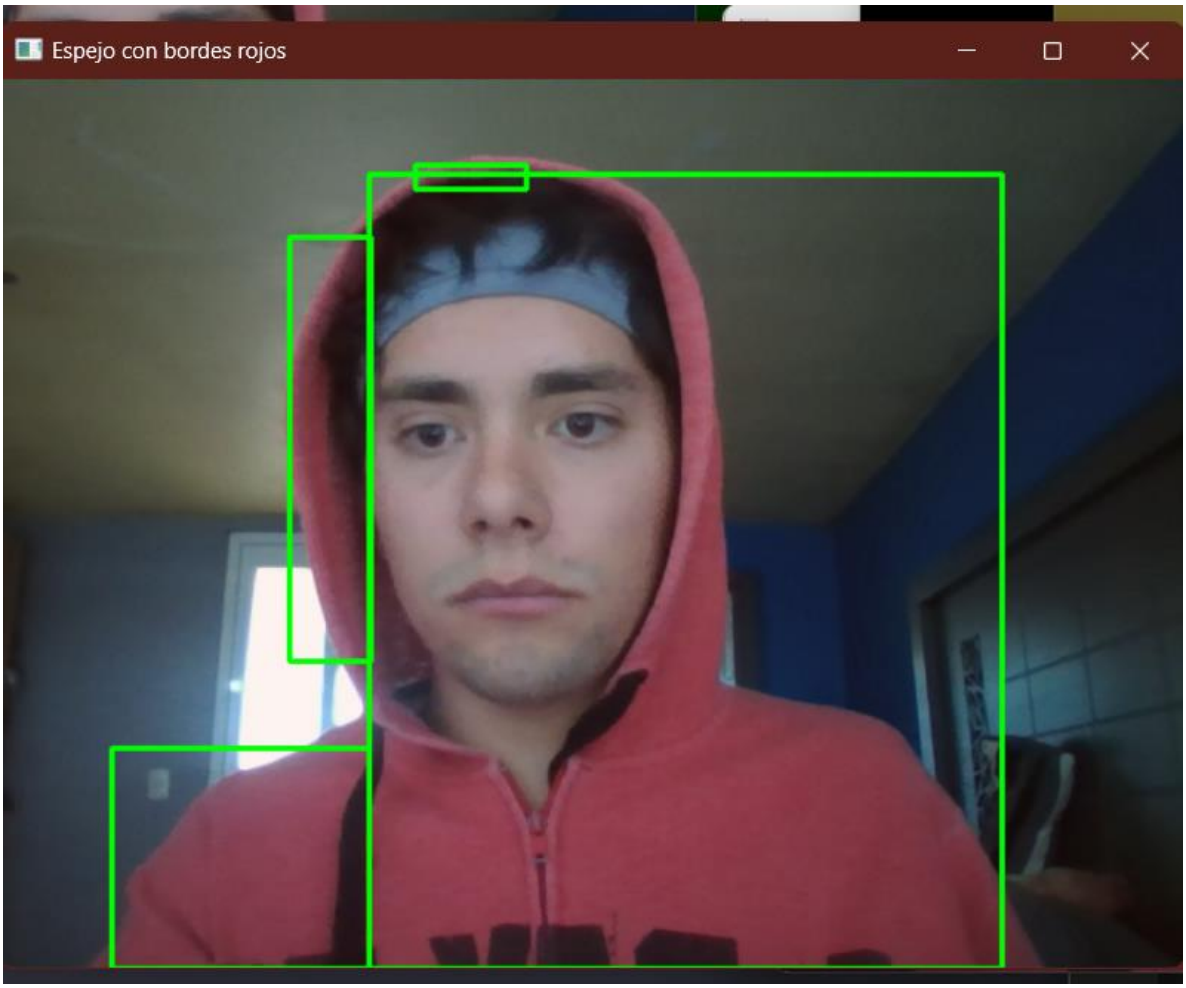


6.-Capa green con mascara:



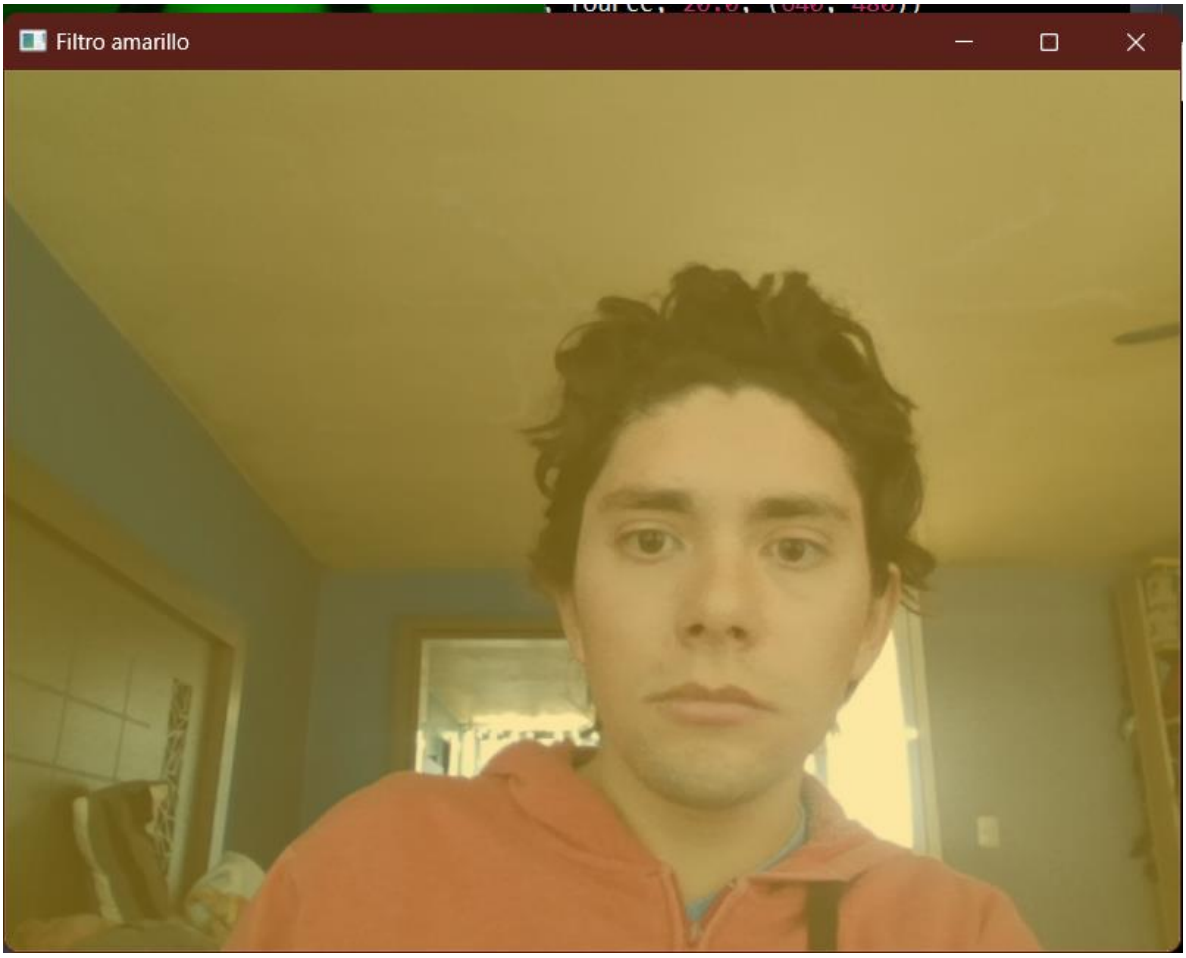

```
# Crear una máscara circular del mismo tamaño que la imagen
mask = np.zeros(frame.shape[:2], dtype=np.uint8)
center = (int(frame.shape[1]/2), int(frame.shape[0]/2))
radius = int(min(frame.shape[1], frame.shape[0]) / 3)
cv2.circle(mask, center, radius, (255, 255, 255), -1)

# Aplicar la máscara a la imagen de GREEN
green_masked = cv2.bitwise_and(green, green, mask=mask)
```

7.- RED en espejo:



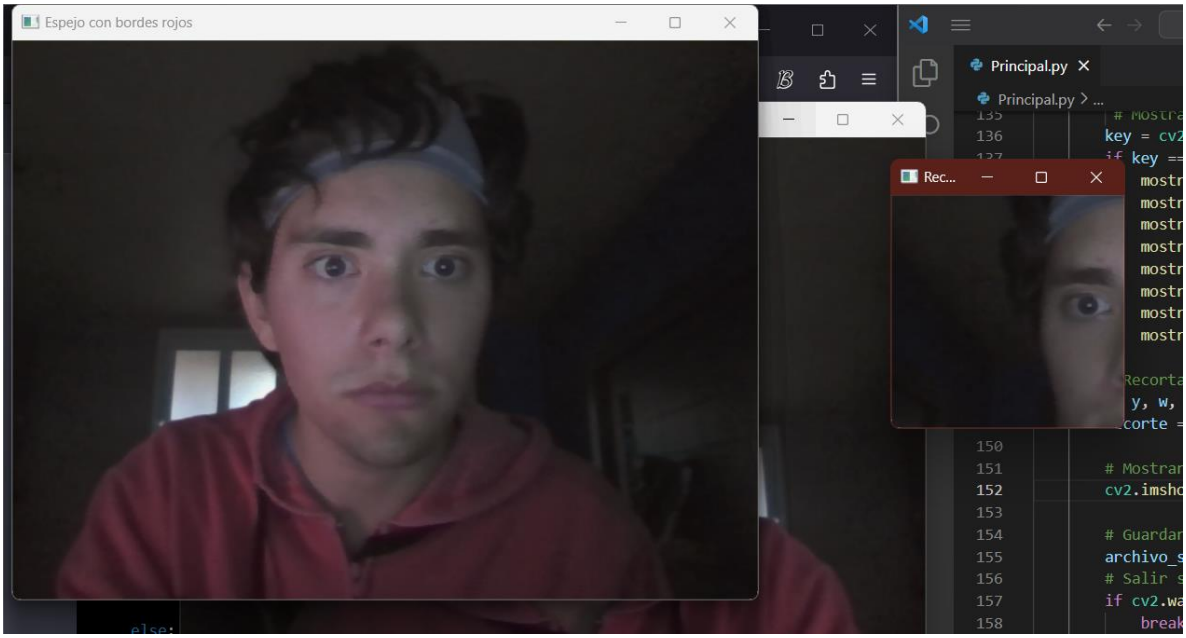

```python
# Dibujar un rectángulo alrededor de los bordes rojos
hsv = cv2.cvtColor(mirror_frame, cv2.COLOR_BGR2HSV)
lower_red = np.array([0, 100, 100])
upper_red = np.array([10, 255, 255])
mask1 = cv2.inRange(hsv, lower_red, upper_red)
lower_red = np.array([160, 100, 100])
upper_red = np.array([179, 255, 255])
mask2 = cv2.inRange(hsv, lower_red, upper_red)
mask = cv2.bitwise_or(mask1, mask2)
contours, hierarchy = cv2.findContours(mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
for cnt in contours:
    x, y, w, h = cv2.boundingRect(cnt)
    if w*h > 500:
        cv2.rectangle(mirror_frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

# Mostrar la imagen con efecto espejo en una nueva ventana
cv2.imshow('Espejo con bordes rojos', mirror_frame)
```

8.- Imagen original con filtro sepia:



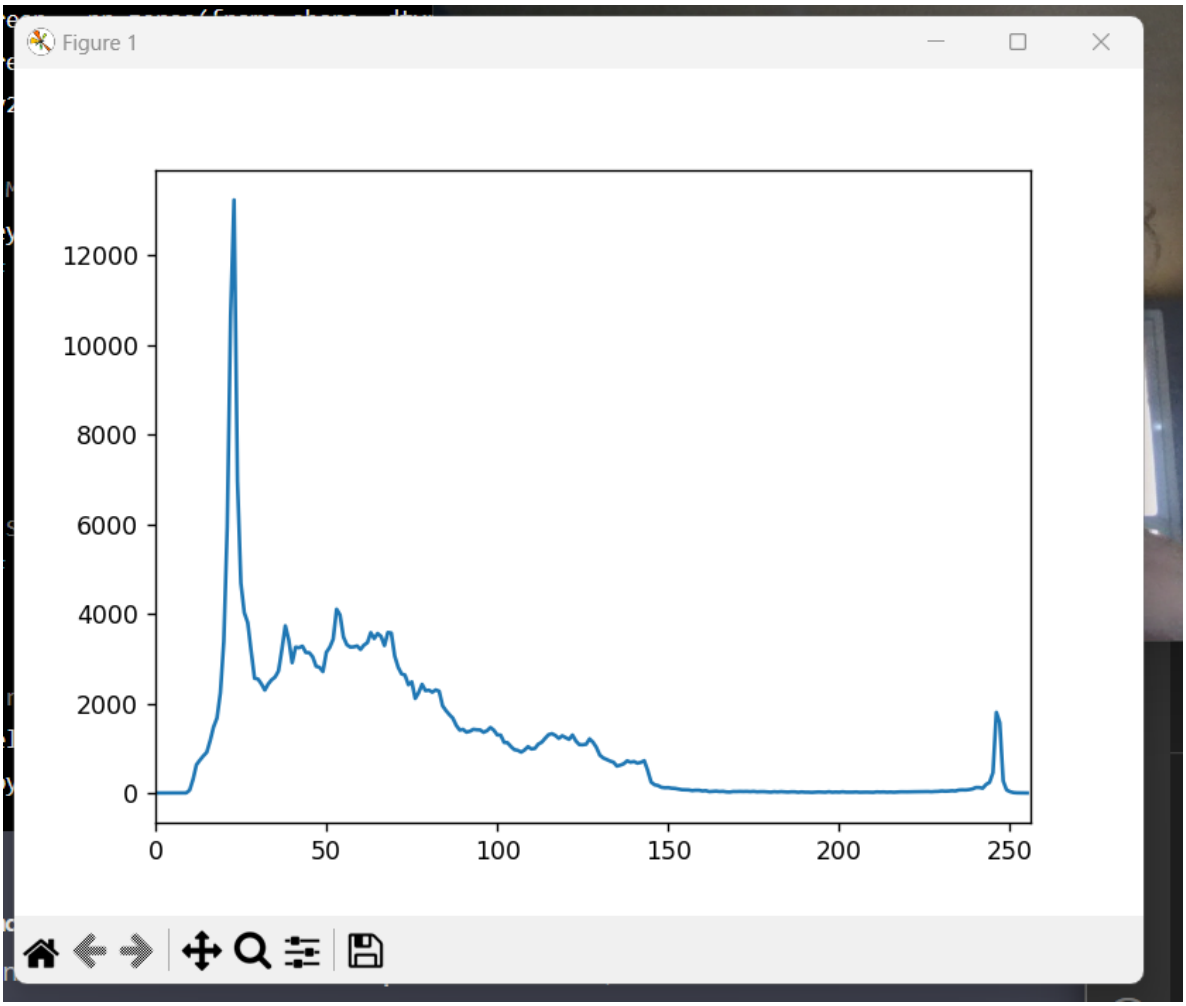

```
    # Aplicar filtro sepia amarillo
sepia_amarillo = np.zeros(frame.shape, dtype=np.uint8)
sepia_amarillo[:,:,0] = 25
sepia_amarillo[:,:,1] = 204
sepia_amarillo[:,:,2] = 255
sepia_frame = cv2.addWeighted(frame, 0.6, sepia_amarillo, 0.4, 0)

# Mostrar la vista con filtro sepia amarillo en la ventana correspondiente
cv2.imshow('Filtro amarillo', sepia_frame)
```

9.-Segmento grabcut:




```
# Recortar una porción de la imagen caricatura
x, y, w, h = 100, 100, 200, 200 # coordenadas y tamaño del re
recorte = frame[y:y+h, x:x+w]

# Mostrar la imagen recortada en una nueva ventana
cv2.imshow('Recorte', recorte)
```

10.- Histograma:



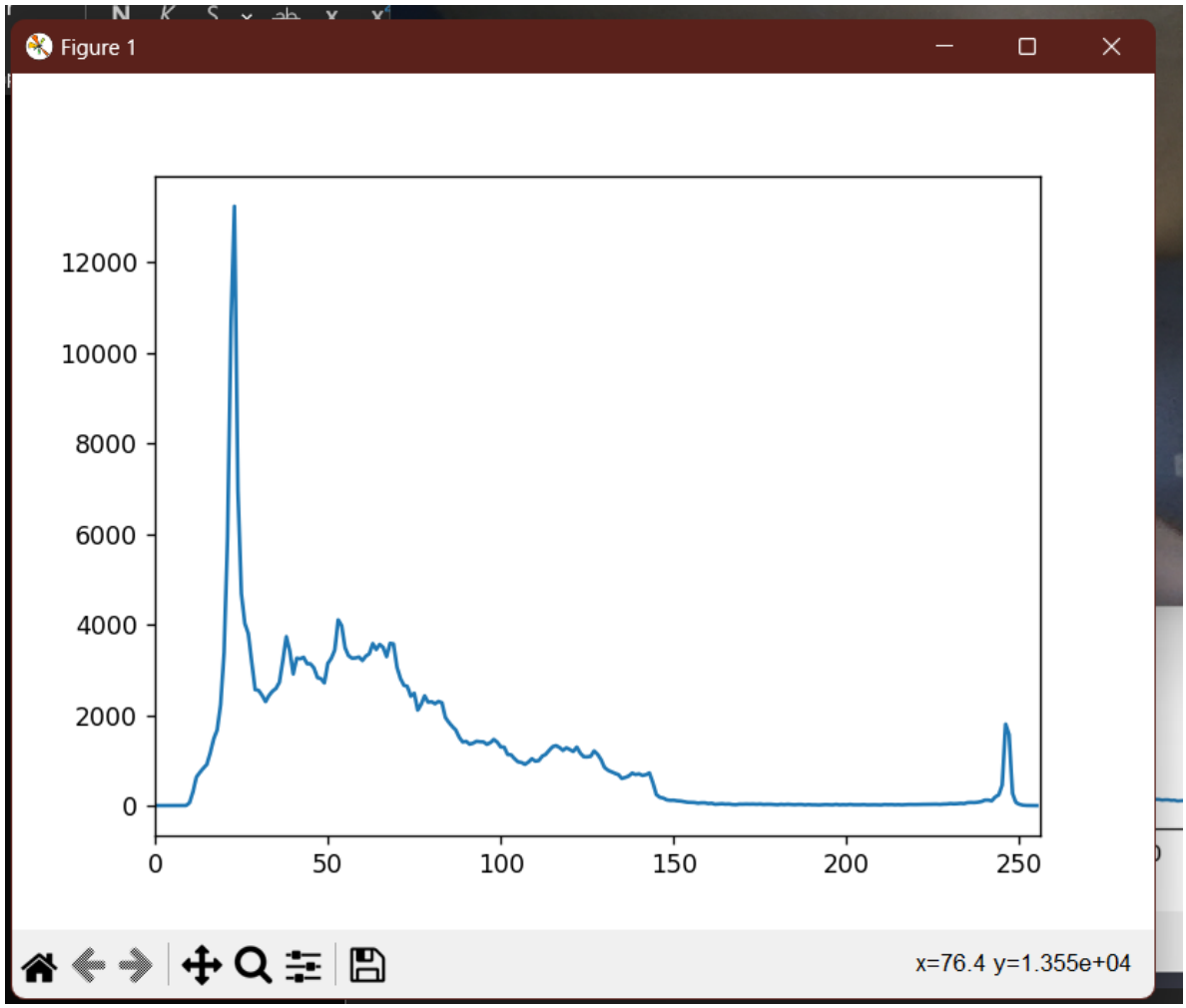


Figure 1
12000
10000
8000
6000
4000
2000
0
0
50
100
150
200
250
x=76.4 y=1.355e+04

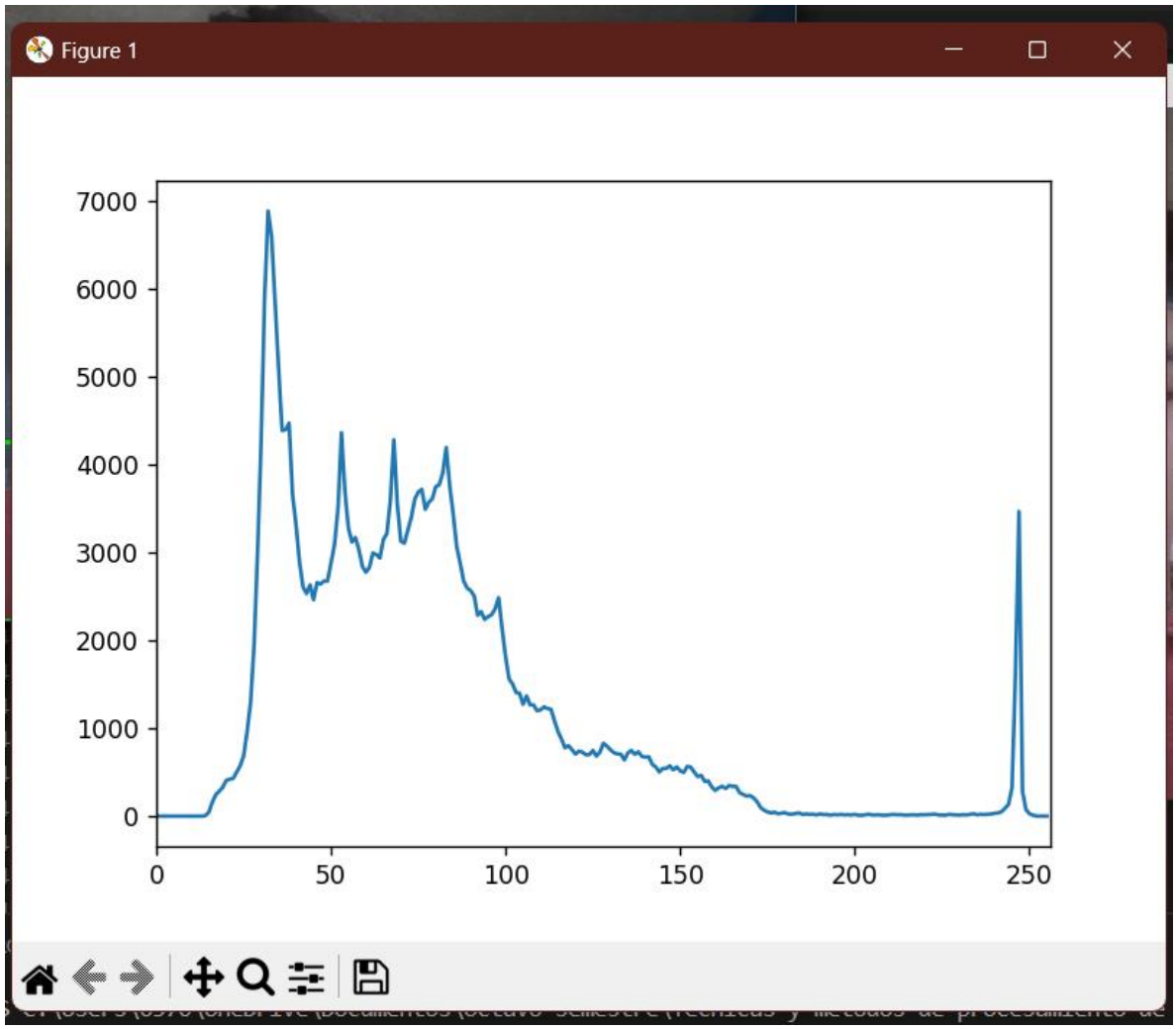


Figure 1
7000
6000
5000
4000
3000
2000
1000
0
0
50
100
150
200
250

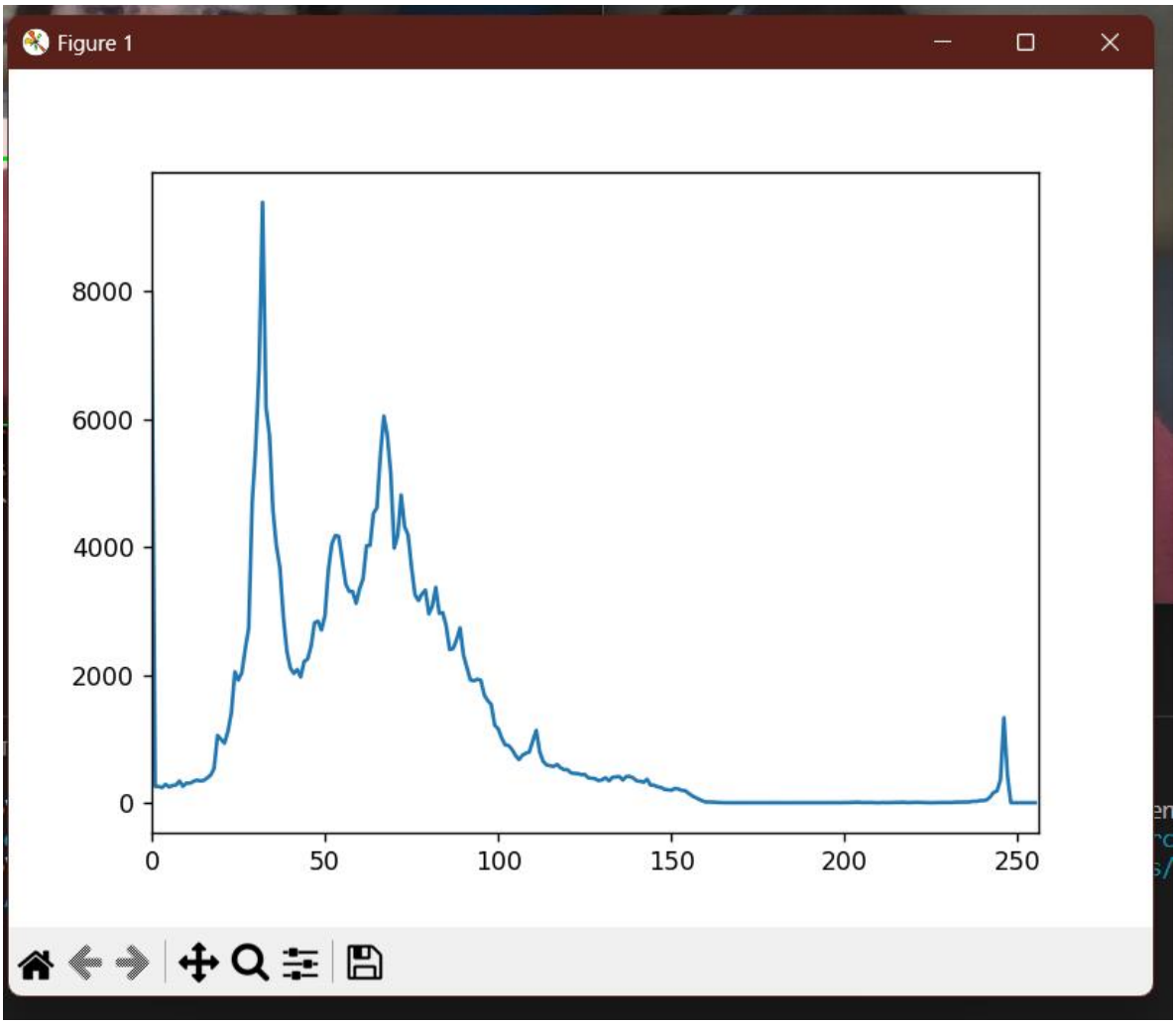
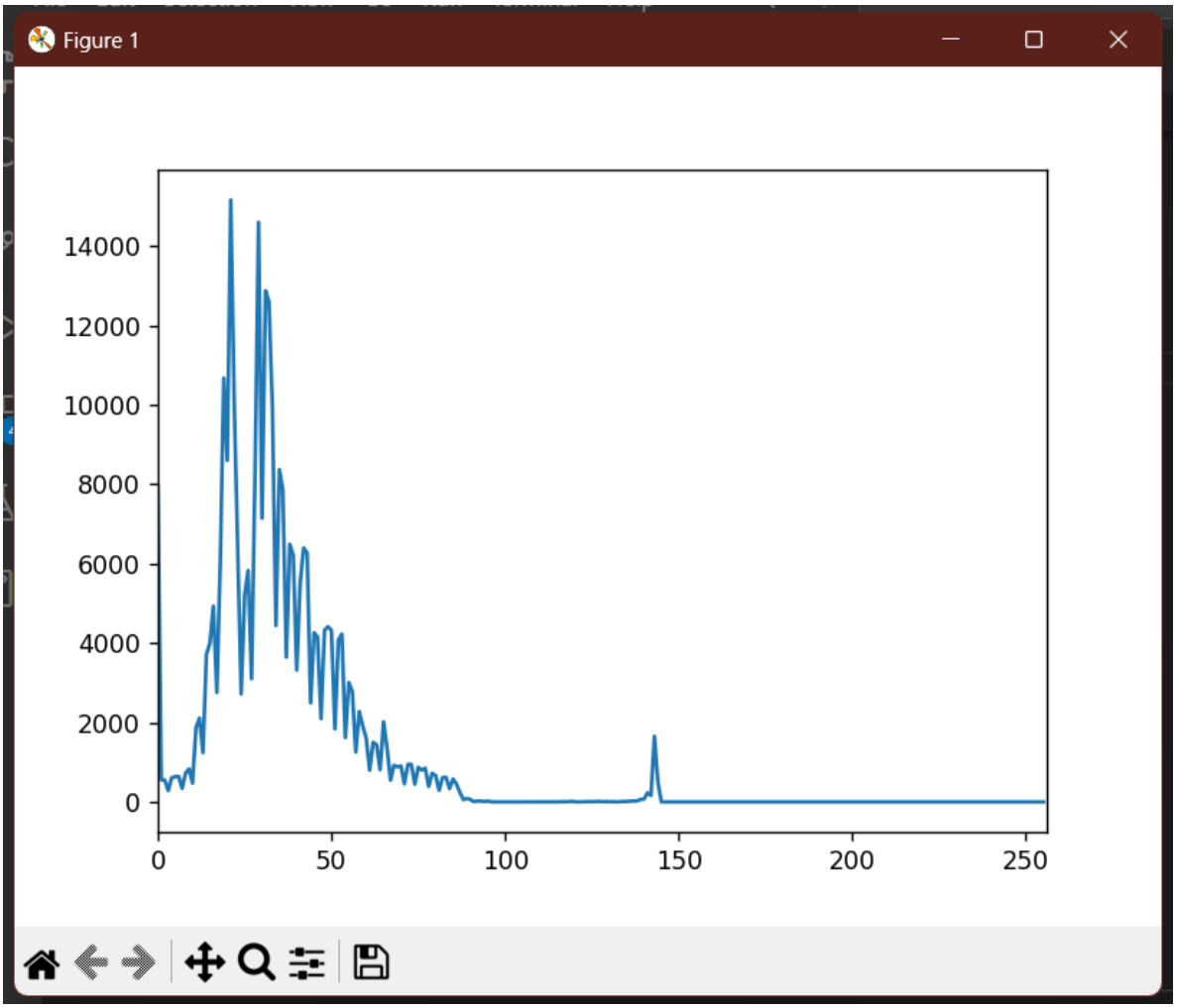


Figure 1
8000
6000
4000
2000
0
0
50
100
150
200
250




```
 # Mostrar histograma de la ventana correspondiente al presionar la tecla 'h'
key = cv2.waitKey(1) & 0xFF
if key == ord('h'):
    mostrar_histograma(frame)
    mostrar_histograma(cv2.flip(frame, 1))
    mostrar_histograma(caricatura)
    mostrar_histograma(green)
    mostrar_histograma(circular_mask)
    mostrar_histograma(mirror_frame)
    mostrar_histograma(sepia_frame)
    mostrar_histograma(masked_green)
```

```
import matplotlib.pyplot as plt
```

Código completo:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt


# Capturar video de la cámara
captura = cv2.VideoCapture(0)

# Configurar codec y formato de video
fourcc = cv2.VideoWriter_fourcc(*'XVID')
archivo_salida = cv2.VideoWriter('output.avi', fourcc, 20.0, (640, 480))

# Bandera para activar el efecto espejo
espejo = False

# Crear ventana para la vista espejo
cv2.namedWindow('Espejo', cv2.WINDOW_NORMAL)

# Crear ventana para la vista de caricatura
cv2.namedWindow('Caricatura', cv2.WINDOW_NORMAL)

# Crear ventana para la vista en canal GREEN
cv2.namedWindow('Canal verde', cv2.WINDOW_NORMAL)


# Función para mostrar histograma de una imagen
def mostrar_histograma(imagen):
    # Convertir imagen a escala de grises
    imagen_gris = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)
    # Calcular histograma
    hist = cv2.calcHist([imagen_gris], [0], None, [256], [0, 256])
    # Mostrar histograma en una ventana
    plt.plot(hist)
    plt.xlim([0, 256])
    plt.show()


while (captura.isOpened()):
    # Capturar cada frame del video
    ret, frame = captura.read()

    if ret == True:
        # Verificar si se debe aplicar el efecto espejo
        if espejo:
```

```
            frame = cv2.flip(frame, 1)

        # Obtener el canal verde y dejar los otros dos canales en cero
        green = np.zeros(frame.shape, dtype=np.uint8)
        green[:, :, 1] = frame[:, :, 1]

        # Mostrar la vista en canal GREEN en la ventana correspondiente
        cv2.imshow('Caricatura verde', green)

        # Aplicar efecto de caricatura
        caricatura = cv2.stylization(frame, sigma_s=60, sigma_r=0.07)

        # Mostrar el video en una ventana
        cv2.imshow('Video', frame)

        # Mostrar la vista espejo en la ventana correspondiente
        cv2.imshow('Espejo', cv2.flip(frame, 1))

        # Mostrar la vista de caricatura en la ventana correspondiente
        cv2.imshow('Caricatura', caricatura)

        # Mostrar la vista en canal GREEN en la ventana correspondiente
        cv2.imshow('Canal verde', green)

        # Obtener el canal verde y dejar los otros dos canales en cero
        green = np.zeros(frame.shape, dtype=np.uint8)
        green[:, :, 1] = caricatura[:, :, 1]

        # Mostrar la vista en canal GREEN en la ventana correspondiente
        cv2.imshow('Caricatura verde', green)

        # Crear una máscara circular para el canal verde
        center_coordinates = (int(green.shape[1]/2), int(green.shape[0]/2))
        radius = int(min(green.shape[1], green.shape[0])/2)
        circular_mask = np.zeros(green.shape[:2], np.uint8)
        cv2.circle(circular_mask, center_coordinates, radius, 255, -1)

        # Aplicar la máscara circular al canal verde
        masked_green = cv2.bitwise_and(green, green, mask=circular_mask)

        # Mostrar la vista en canal GREEN con máscara en la ventana
correspondiente
        cv2.imshow('Canal verde con mascara', masked_green)

        # Aplicar umbral a la imagen caricatura
```

```
        umbralizada = cv2.cvtColor(caricatura, cv2.COLOR_BGR2GRAY)
        _, umbralizada = cv2.threshold(
            umbralizada, 127, 255, cv2.THRESH_BINARY)

        # Mostrar la imagen umbralizada en la ventana correspondiente
        cv2.imshow('Umbralizada', umbralizada)

        # Aplicar filtro sepia amarillo
        sepia_amarillo = np.zeros(frame.shape, dtype=np.uint8)
        sepia_amarillo[:, :, 0] = 25
        sepia_amarillo[:, :, 1] = 204
        sepia_amarillo[:, :, 2] = 255
        sepia_frame = cv2.addWeighted(frame, 0.6, sepia_amarillo, 0.4, 0)

        # Mostrar la vista con filtro sepia amarillo en la ventana
correspondiente
        cv2.imshow('Filtro sepia', sepia_frame)

        # Aplicar efecto espejo a la imagen original
        mirror_frame = cv2.flip(frame, 1)

        # Dibujar un rectángulo alrededor de los bordes rojos
        hsv = cv2.cvtColor(mirror_frame, cv2.COLOR_BGR2HSV)
        lower_red = np.array([0, 100, 100])
        upper_red = np.array([10, 255, 255])
        mask1 = cv2.inRange(hsv, lower_red, upper_red)
        lower_red = np.array([160, 100, 100])
        upper_red = np.array([179, 255, 255])
        mask2 = cv2.inRange(hsv, lower_red, upper_red)
        mask = cv2.bitwise_or(mask1, mask2)
        contours, hierarchy = cv2.findContours(
            mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
        for cnt in contours:
            x, y, w, h = cv2.boundingRect(cnt)
            if w*h > 500:
                cv2.rectangle(mirror_frame, (x, y), (x+w, y+h), (0, 255, 0),
2)

        # Mostrar la imagen con efecto espejo en una nueva ventana
        cv2.imshow('Espejo con bordes rojos', mirror_frame)

        # Mostrar histograma de la ventana correspondiente al presionar la
tecla 'h'
        key = cv2.waitKey(1) & 0xFF
        if key == ord('h'):
```

```
            mostrar_histograma(frame)
            mostrar_histograma(cv2.flip(frame, 1))
            mostrar_histograma(caricatura)
            mostrar_histograma(green)
            mostrar_histograma(circular_mask)
            mostrar_histograma(mirror_frame)
            mostrar_histograma(sepia_frame)
            mostrar_histograma(masked_green)

        # Recortar una porción de la imagen caricatura
        x, y, w, h = 100, 100, 200, 200  # coordenadas y tamaño del recorte
        recorte = frame[y:y+h, x:x+w]

        # Mostrar la imagen recortada en una nueva ventana
        cv2.imshow('Recorte', recorte)

        # Guardar el video en archivo de salidaq
        archivo_salida.write(frame)
        # Salir si se presiona la tecla 'q'
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
        # Activar el efecto espejo si se presiona la tecla 's'
        elif cv2.waitKey(1) & 0xFF == ord('s'):
            espejo = not espejo
    else:
        break

# Liberar recursos y cerrar ventanas
captura.release()
archivo_salida.release()
cv2.destroyAllWindows()
```

Video guardado en carpeta: