Tarea realizada por Noel Padrón Jiménez y Joel Rodríguez González

Proyecto de Visión por Computador

Descripción:

Este proyecto implementa diversas tareas de procesamiento de imágenes utilizando técnicas de visión por computador en Python, principalmente a través de la librería OpenCV. A lo largo del proyecto, se desarrollan tareas como la detección de bordes, conversión de imágenes a escala de grises, análisis de la intensidad de píxeles, y captura de video en tiempo real para aplicar filtros.

Requisitos. Para ejecutar este proyecto, es necesario tener instalados los siguientes paquetes:

OpenCV: Para la captura y procesamiento de imágenes.

NumPy: Para la manipulación de matrices.

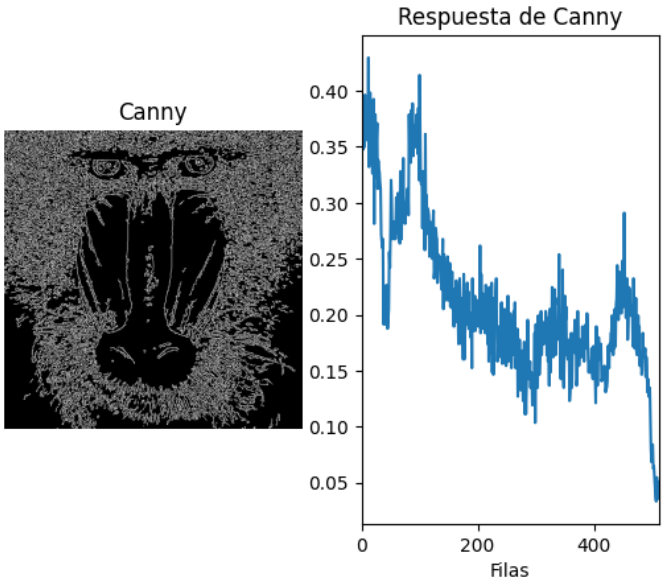
Matplotlib: Para la visualización de imágenes.

Funcionalidades Implementadas:

1. Conteo de Píxeles Blancos por Filas en la Imagen Canny.

   Descripción: Se realiza el conteo de píxeles blancos (255) en la imagen resultante de la transformación de Canny, calculando la cantidad por filas en lugar de por columnas. Procedimiento: La imagen es transformada utilizando Canny. Se realiza la cuenta de los píxeles blancos para cada fila y se calcula el valor máximo, maxfil. Se identifican las filas con un número de píxeles blancos mayor o igual a 0.95 * maxfil. Se dibujan líneas en la imagen para resaltar las filas seleccionadas.

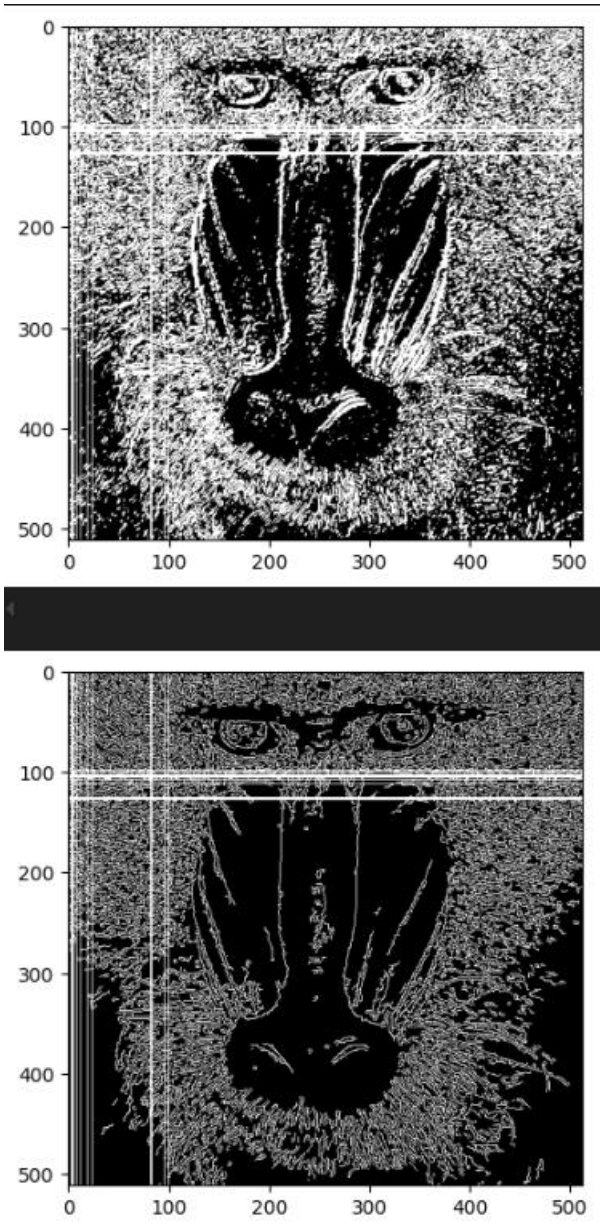
   Resultado: Se visualiza la imagen con las filas destacadas, junto con una gráfica que muestra el porcentaje de píxeles blancos por fila.



2. Umbralizado y Conteo por Filas y Columnas en la Imagen Sobel.

Descripción: Se aplica un umbral a la imagen resultante de Sobel para convertirla a binaria, seguido del conteo de píxeles blancos por filas y columnas. Procedimiento: Se aplica Sobel para calcular las derivadas en los ejes X e Y de la imagen en escala de grises. Se umbraliza la imagen de Sobel para obtener una imagen binaria. Se realiza el conteo de píxeles blancos por filas y columnas en la imagen umbralizada, calculando el valor máximo para cada caso. Se identifican las filas y columnas con un número de píxeles blancos mayor o igual a 0.95 * maxfil. Se remarcan dichas filas y columnas en la imagen umbralizada utilizando líneas.

Comparación: Se comparan visualmente los resultados obtenidos con Sobel y Canny para analizar las diferencias en la detección de bordes y la distribución de píxeles blancos.

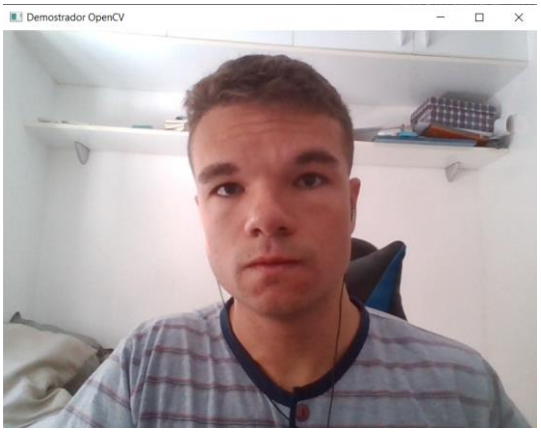


3. Demostrador Interactivo

Se implementa un demostrador que captura video en tiempo real desde la webcam, permitiendo cambiar entre diferentes modos de visualización:

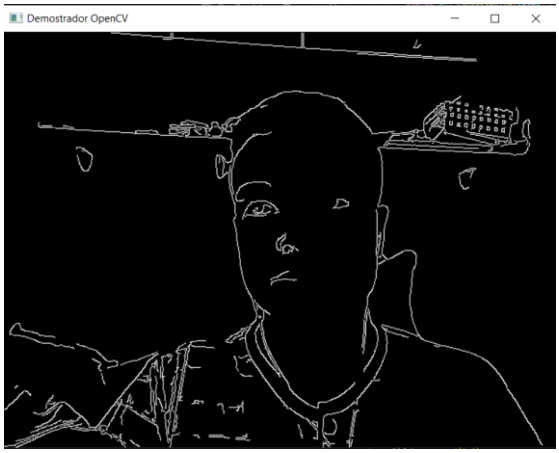
Modos disponibles:

o: Imagen Original

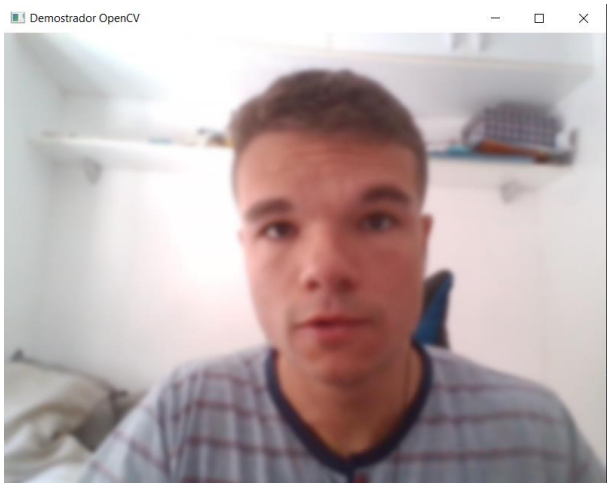


g: Escala de Grises

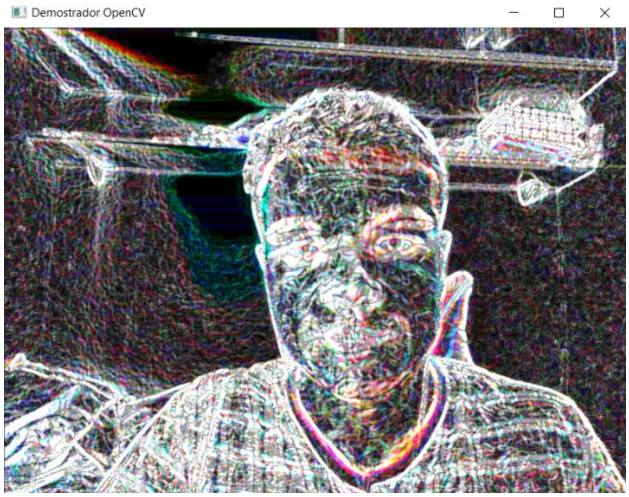
c: Bordes con Canny



b: Desenfoque Gaussiano

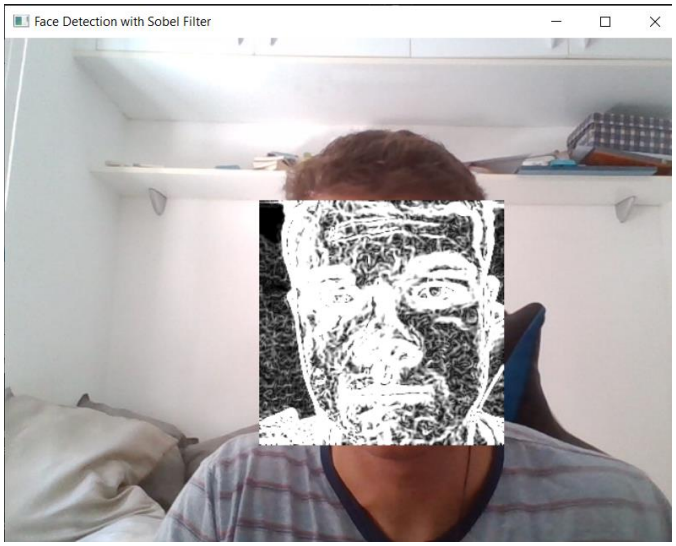


s: Filtro Sobel



4. Detección de Rostros y Aplicación de Filtros

Se implementa un sistema que detecta rostros en tiempo real utilizando el clasificador entrenado y aplica el filtro Sobel para resaltar los bordes de la cara detectada.



Para realizar esta última tarea nos inspiramos en la del vídeo de Virtual Air Guitar, pero en lugar de reconocer las manos, quisimos que el programa reconozca la cara y le aplique un filtro, para detectar la cara empleamos la siguiente línea:

face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')

Que lo que hace es cargar un modelo preentrenado para la detección frontal de rostros faciales.

Nos fijamos del siguiente vídeo:

https://www.youtube.com/watch?v=J1jlm-I1cTs&t=389s

Para ejecutar el proyecto, asegúrate de tener conectada una cámara web.

Ejecuta el script de Python que corresponde a la funcionalidad que deseas probar.

Usa la tecla ESC o “Q” según corresponda para detener la ejecución y cerrar las ventanas de visualización.