Universidad Tecnológica Nacional

FRRO

Cátedra: Soporte
Trabajo Práctico Integrador
Detección de prendas de color de personas con OpenCV

4º Año de Ingeniería en Sistemas de Información
Comisión 405

Docentes:
Ing. Mario Castagnino
Ing. Juan Ignacio Torres

Alumnos:
50020 - Cravero, Pablo - pablocraveroutn@gmail.com
49497 - Gini, Luca - luca.giniclc@gmail.com
49742 - Gutierrez, Luisina - luisinagutierrez2618@gmail.com
50180 - Pérez Fontela, Simón - ipcsimon.tel@gmail.com

Fecha entrega: **15/10/2024**

Abstract

This project aims to develop an automated real-time detection system using Python and OpenCV to identify individuals based on their clothing from security camera footage in Rosario. Additionally, it stores all detections in a local database and folders for future forensic analysis, enhancing public safety and supporting criminal investigations.

Índice

Narrativa

Dado el alarmante aumento de crímenes en la ciudad de Rosario, se ha decidido desarrollar un sistema de detección automática, cuya función principal es identificar personas en función de las prendas que están utilizando. Este sistema tiene como objetivo la realización de un análisis forense preciso a partir de los videos de las cámaras de seguridad instaladas en distintos puntos estratégicos de la ciudad para permitir una respuesta rápida y eficaz de las autoridades locales.

Una vez que una cámara detecta la presencia de una persona con el color de ropa buscado, se almacena inmediatamente en una base de datos local en la raíz del proyecto para su futura referencia y análisis forense, lo que permitirá tener un registro detallado y accesible de cada incidente reportado.

Para lograr esta funcionalidad, el sistema empleará un modelo de aprendizaje automático basado en una red neuronal pre-entrenada para la identificación precisa de personas. Este modelo analiza las imágenes captadas por las cámaras, y con un nivel de confianza definido previamente, es capaz de reconocer y delimitar las personas detectadas. En cuanto el sistema identifica coincidencias con los parámetros establecidos por el usuario, captura automáticamente un video de la persona en cuestión.

Requerimientos

Funcionales

- El sistema debe ser capaz de distinguir personas bajo ciertos parámetros.
- El sistema debe almacenar cada una de las detecciones realizadas.
- El sistema debe delimitar en una imagen o video la detección realizada.
- El sistema debe ser capaz de persistir las grabaciones de detección y sus datos en una base de datos local, para posterior análisis.
- El sistema debe ser capaz de manejar un flujo de personas considerable y tratar debidamente las situaciones donde haya muchas personas con prendas del mismo color.

No funcionales

- El sistema debe funcionar en PCs con Windows 10 y 11 como su sistema operativo.
- El sistema debe ser desarrollado utilizando Python y OpenCV.

Stack tecnológico

Para la implementación de este proyecto de análisis forense, se ha utilizado el lenguaje de programación Python en su versión 3.12.2, que ofrece una amplia gama de bibliotecas y herramientas ideales para el procesamiento de imágenes y videos en tiempo real. Entre los módulos y tecnologías más relevantes se encuentran:

- OpenCV: Una biblioteca poderosa para la manipulación de imágenes y videos. OpenCV facilita el acceso a las cámaras de seguridad y permite realizar tareas avanzadas de procesamiento de imágenes, como la segmentación de colores y el seguimiento de objetos.
- YOLO (You Only Look Once): Un algoritmo de detección de objetos que utiliza técnicas de aprendizaje profundo (Deep Learning). YOLO es conocido por su capacidad para realizar detecciones en tiempo real con alta precisión. En este proyecto, YOLO se utiliza para identificar y localizar personas en los videos que visten ropa del color solicitado.

Ambos componentes trabajan de manera conjunta para analizar los videos, identificar a las personas usando el color especificado y generar clips relevantes que incluyen momentos antes, durante y después de la aparición de dichas personas en escena.

Reglas de negocio

El sistema de análisis forense se rige por un conjunto de reglas que aseguran la precisión y eficiencia en la detección de colores y la gestión de los datos asociados. Las principales reglas de negocio implementadas son:

- Detección sin importar la duración: Si el sistema detecta el color ingresado en una o varias personas, este debe ser señalado y procesado sin importar la duración de su aparición en el video.
- Identificación visual: Cada vez que se detecta el color especificado, el sistema lo remarca visualmente en el video, permitiendo una rápida identificación de la persona o personas que lo portan.
- Almacenamiento de videos relevantes: Para cada detección de color realizada, el sistema almacena un video de 5 segundos previos a la detección y 10 segundos posteriores, generando un resumen que facilita la visualización de los momentos clave.
- Filtrado de videos: No se almacenan videos o imágenes en los que no se detecta el color solicitado en una persona, optimizando así el uso de espacio en el sistema y manteniendo solo el material relevante.
- Respaldo periódico de videos: Se realizan backups periódicos de los videos almacenados en la base de datos. Esto no solo asegura la integridad de los datos, sino que también permite parametrizar la eficiencia del modelo, al facilitar el acceso y análisis de las detecciones pasadas.

Caso de uso

Detección y almacenamiento de video basado en el color de vestimenta de las personas.

Nivel	Estructura	Alcance	Caja	Instanciación	Interacción
Resumen	Sin estructurar	Sistema	Negra	Real	Semántico

Meta del caso de uso: Determinar y almacenar un video que muestre a las personas detectadas usando el color ingresado.

Actor primario: Operador del sistema.

Otros: cámaras de vigilancia.

Precondiciones (de sistema):

- El sistema de detección está cargado y operativo en el equipo.
- Las cámaras de vigilancia están encendidas y transmiten video en tiempo real.

Disparador: el operador inicia el equipo con el sistema de detección.
Camino básico:

1. El operador enciende el equipo y carga el sistema de detección.

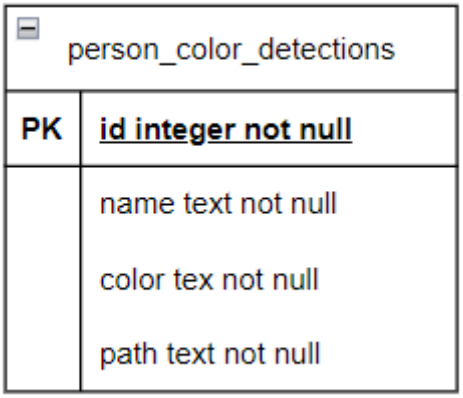
2. El operador ingresa el color de la vestimenta que desea que el sistema detecte en las personas.
3. Las cámaras de vigilancia capturan a las personas en tiempo real y el sistema comienza el análisis utilizando un porcentaje de confianza basado en el modelo de detección.
4. El sistema analiza los videos en busca de personas portando el color seleccionado.
5. Si una o más personas son detectadas con el color especificado, el sistema guarda la detección en la base de datos, registrando la fecha y hora del evento junto con un enlace a un video que incluye:
    - 5 segundos previos a la detección.
    - 10 segundos posteriores a la detección.

Los pasos 3 - 5 se repiten hasta que ya no se necesite de los servicios del sistema.

Caminos alternativos.
5.1 *<durante> Si el sistema no detecta personas con el color solicitado, no se almacena ningún video ni dato. El sistema sigue procesando en tiempo real hasta que ocurra una detección.

Postcondiciones (de sistema)
Éxito: El sistema ha detectado y almacenado un video relevante con un resumen del evento.
Fracaso: No se detectaron personas usando el color especificado durante el tiempo de funcionamiento del sistema.
Éxito alternativo: --

Modelo de dominio



Debido a los objetivos del proyecto, la base de datos está diseñada para almacenar exclusivamente los videos relacionados con las detecciones realizadas por el sistema. Cada video almacenado contiene información clave para su identificación y gestión, incluyendo:

- ID único: Un identificador único que permite diferenciar cada detección de las demás.
- Nombre: El nombre del archivo, asociado al color detectado en el evento.

- Color detectado: Especificación del color de la vestimenta que fue objeto de la detección.
- Ruta del video: La ubicación o enlace donde se almacena el video que contiene las imágenes de una o más personas usando el color especificado.

Esta estructura garantiza que el sistema pueda gestionar eficientemente cada detección, facilitando la recuperación rápida y precisa de los videos relevantes para análisis posteriores.

Imágenes

Interfaz de Usuario

Imagen 1: menú principal.

En la imagen 1 podemos observar el menú principal del programa, el cual solicita que seleccione en color por el cual se va a filtrar a las personas y el video sobre el cual quiere realizar el análisis forense. A su vez, permite ver videos guardados en la base de datos.

Selección de color y de video

Imagen 2: selección.

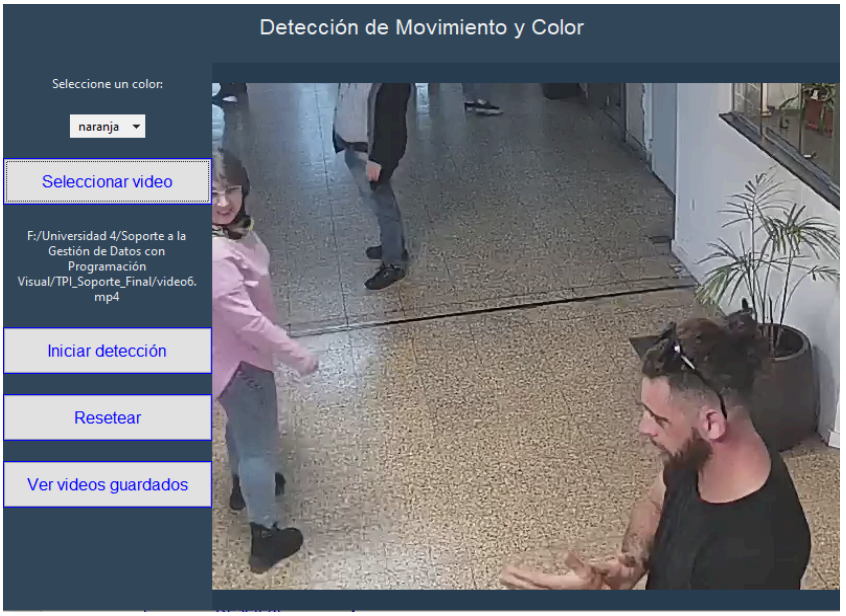


Imagen 3: reproducción del video.

Una vez seleccionado el color como el vídeo, comenzará a mostrarse del lado derecho de la pantalla una preview del video en cuestión (como puede observarse en la imagen 3). Al seleccionar “Iniciar detección”, veremosº cómo las personas son detectadas como tales con el porcentaje de confiabilidad correspondiente.

Acceso a videos guardados

ID	Nombre	Color	Ruta
2	20241009_163810.mp4	naranja	naranja\20241009_163810.mp4
3	20241010_225318.mp4	naranja	naranja\20241010_225318.mp4

Imagen 4: base de datos.

Cuando el sistema detecta a una persona con el color solicitado, el mismo graba un video desde 5 segundos anteriores a la detección del color hasta 10 segundos después de la misma.
El mismo se almacena en una base de datos, cuyos campos podemos observar en la imagen 4 siendo el ID la clave primaria y única, pero contando además con el nombre, que es la fecha y hora de la grabación, el color detectado y la ruta al video.

Reproducción de videos guardados: Antes y después de la detección.

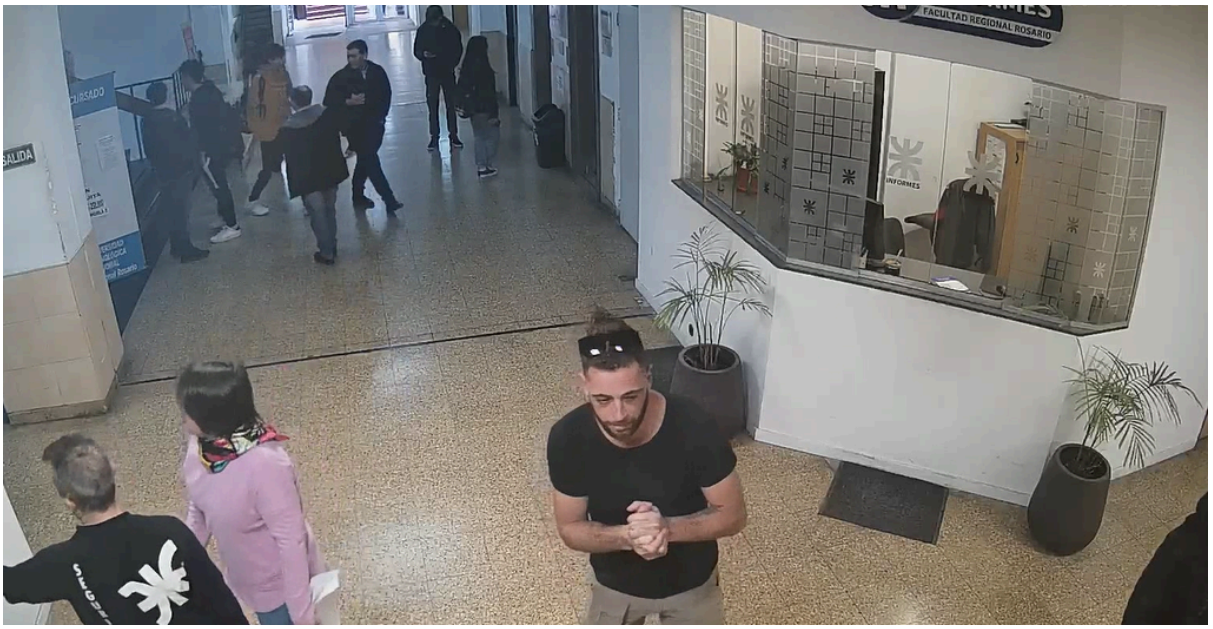


imagen 5: captura del video sin detección del color seleccionado.

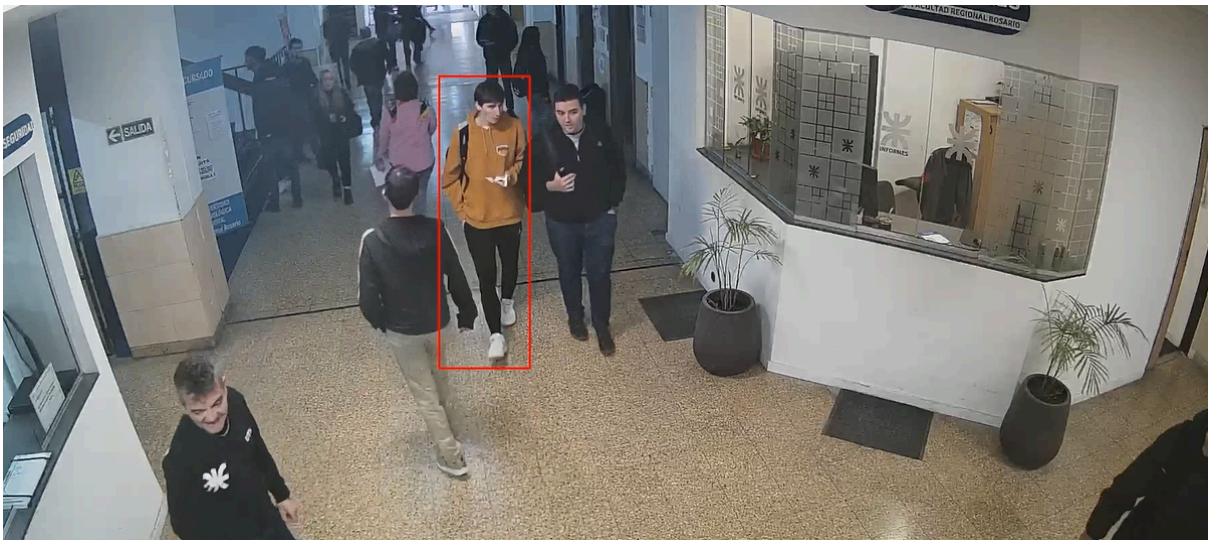


imagen 6: captura del video ante la detección del color seleccionado.

Ambas imágenes (5 y 6) fueron obtenidas del video de grabación ante la detección del color naranja, podemos apreciar que además, para una mayor claridad, cuando la persona entra en el recuadro de detección es la única que es resaltada, permitiendo la distinción de la misma entre las demás, facilitando el análisis forense de detección.

Conclusión

El proyecto "Detección de prendas de color de personas con OpenCV" se presenta como una herramienta innovadora que busca contribuir a la seguridad pública en la ciudad de Rosario. A través del uso de Python y la librería OpenCV, se ha logrado desarrollar un sistema eficiente para la identificación automática de personas basándose en el color de su vestimenta, utilizando imágenes y videos capturados por cámaras de seguridad.

El sistema implementa algoritmos avanzados de detección, como YOLO (You Only Look Once), que permiten realizar el procesamiento en tiempo real, asegurando una rápida respuesta ante posibles situaciones de interés. Además, el sistema almacena los videos de las detecciones en una base de datos local, proporcionando un valioso recurso para análisis forenses posteriores.

Durante el desarrollo del proyecto, el equipo adquirió valiosos conocimientos sobre el procesamiento de imágenes, la integración de algoritmos de machine learning y la gestión eficiente de bases de datos locales. Aprendimos a manejar desafíos como la detección simultánea de múltiples personas y la segmentación de colores en distintos entornos lumínicos. Este proyecto nos permitió aplicar y reforzar habilidades clave para el desarrollo de software complejo.

Si bien el sistema ha demostrado ser eficaz, consideramos que una posible mejora futura sería optimizar el uso de recursos en equipos con menor capacidad de procesamiento. Adicionalmente, la implementación de técnicas más avanzadas de seguimiento de objetos podría mejorar la precisión en escenarios con alta concurrencia de personas.

En resumen, este trabajo integrador no solo muestra la capacidad técnica del equipo para desarrollar soluciones complejas en entornos de procesamiento de imágenes y videos, sino que también presenta una propuesta con potencial de aplicación en el mundo real, demostrando un claro enfoque hacia la resolución de problemas de seguridad y vigilancia en entornos urbanos.

Bibliografía

Documentación de librerías

Las librerías utilizadas en el proyecto son:

- Tkinter: Es la interfaz por defecto de Python para el kit de herramientas de GUI Tk. Permite implementar widgets y clases de Python. Sus principales virtudes son su rapidez y que permite desarrollar ventanas, botones, cuadros de texto y otros elementos interactivos. En el programa, se utilizó para realizar la interfaz. Específicamente, se hizo uso de:
    - Ttk: Es una nueva familia de widgets que proveen una atractiva apariencia en comparación a la de los widgets clásicos, permitiendo usar temas modernos, botones, etiquetas, cuadros de texto, etc.
    - filedialog: Son cuadros de diálogo por defecto que permiten al usuario especificar un archivo para abrir o guardar. Así, permite al usuario elegir videos desde el sistema de archivos.
    - messagebox: Permite el acceso a cuadros de diálogo estándar de Tkinter. Puede crear pop-ups que muestran mensajes de advertencia, información o errores al usuario

  Documentación: https://docs.python.org/es/3/library/tkinter.html

- OpenCV (cv2): Posee funciones referidas a la detección de objetos y al procesamiento de videos. Entre sus funcionalidades, tenemos:
    - Captura de video: Permite leer videos o acceder a cámaras, por lo que es fundamental en este proyecto.

    - Manipular imágenes: Es posible cambiar el tamaño de imágenes o videos, detectar colores en los mismos, realizar ediciones en tiempo real, entre otras. En el proyecto, esta función es muy útil para mostrar a las personas detectadas, resaltando a aquellas que coinciden con el parámetro de color ingresado
    - Detección de movimiento: OpenCV permite filtrar aquellos objetos que se están moviendo. En el programa, esta función fue utilizada para mejorar la precisión con la que se detecta a las personas y analizar el color de su vestimenta.
    - Escritura de vídeo: Refiere al hecho de guardar videos procesados en el disco, función utilizada para el guardado del video de la detección en una base de datos.

  Documentación: https://docs.opencv.org/4.x/

- Numpy: Permite realizar cálculos numéricos, y es especialmente útil para manejar arrays multidimensionales y matrices. En el programa se utiliza al aplicar filtros de color o calcular áreas en movimiento manejando las matrices de píxeles en las imágenes.
  Documentación: https://numpy.org/doc/stable/

- Collections (deque): Permite agregar o quitar elementos de ambos extremos de una cola de manera eficiente. En el programa, se utiliza para generar un buffer que almacena frames del video para luego “escribirlos” al guardar los videos de las detecciones.
  Documentación: https://docs.python.org/3/library/collections.html

- Datetime: Maneja fechas y horas, dando la posibilidad de obtener la fecha y hora actual, formatear fechas o trabajar con intervalos de tiempo. En este caso, se usa para marcar el nombre de cada video con la fecha y hora en que ocurrió la detección, facilitando y optimizando el seguimiento de cuándo ocurrieron los eventos.
  Documentación: https://docs.python.org/3/library/datetime.html

- Os: Permite interactuar con el sistema operativo, dando la posibilidad de manejar archivos y directorios. De este modo, puede crear, eliminar o comprobar existencias de carpetas y archivos en el sistema. En el caso del programa, es utilizado para crear carpetas con los nombres de los colores detectados, manteniendo una organización eficiente acerca de los videos guardados.
  Documentación: https://docs.python.org/3/library/os.html

- PIL (Python Imaging Library): Es una librería para trabajar con imágenes en Python que permite cargar, modificar y guardar imágenes en varios formatos. Se utiliza, en este caso, para convertir los frames capturados en un formato que pueda ser mostrado dentro de la UI. Utilizamos:
    - Image: Permite abrir y manipular imágenes.
    - ImageTk: Convierte las imágenes en un formato que puede ser mostrado en UI.

  Documentación: https://pillow.readthedocs.io/en/stable/

- SQLite3: Es una base de datos que permite interactuar con bases de datos SQLite y no requiere un servidor separado. Se usa para realizar operaciones de bases de datos (crear tablas, insertar registros, hacer consultas), y para gestionar datos persistentes (almacenar información sobre los videos guardados).
  Documentación: https://docs.python.org/3/library/sqlite3.html

Código fuente

Repositorio git
https://github.com/Simon-PF2003/TPI_Soporte.git

main.py

```
import tkinter as tk
from tkinter import filedialog, messagebox, ttk
import cv2
import numpy as np
from collections import deque
from datetime import datetime
import os
from PIL import Image, ImageTk
import sqlite3

conn = sqlite3.connect('videos.db')
cursor = conn.cursor()
cursor.execute('''CREATE TABLE IF NOT EXISTS videos
                (id INTEGER PRIMARY KEY AUTOINCREMENT,
                name TEXT,
                color TEXT,
                path TEXT) ''')
conn.commit()

# Diccionario con rangos más detallados de colores en HSV
color_ranges = {
    "rojo_bajo": [(0, 100, 50), (9, 255, 255)],
    "rojo_alto": [(160, 100, 50), (180, 255, 255)],
    "naranja": [(10, 110, 70), (23, 255, 255)],
    "amarillo": [(20, 70, 70), (33, 255, 255)],
    "verde_oscuro": [(34, 50, 40), (50, 255, 255)],
    "verde_claro": [(51, 50, 40), (85, 255, 255)],
    "azul_claro": [(86, 100, 90), (110, 255, 255)],
    "azul_oscuro": [(111, 100, 90), (130, 255, 255)],
    "violeta": [(131, 100, 100), (145, 255, 255)],
    "rosa_alto": [(145, 50, 50), (160, 255, 255)],
    "rosa": [(0, 40, 100), (10, 60, 255)],
    "rosa_bajo": [(161, 65, 50),(180, 100, 255)],
    "blanco": [(0, 0, 200), (180, 30, 255)],
```

```
    "negro": [(0, 0, 0), (180, 255, 60)],
}

def get_color_ranges(color_name):
    color_name = color_name.lower()
    if color_name.startswith("rojo"):
        lower_red1, upper_red1 = color_ranges["rojo_bajo"]
        lower_red2, upper_red2 = color_ranges["rojo_alto"]
        return (lower_red1, upper_red1, lower_red2, upper_red2, None, None)
    elif color_name.startswith("verde"):
        lower_green1, upper_green1 = color_ranges["verde_oscuro"]
        lower_green2, upper_green2 = color_ranges["verde_claro"]
        return (lower_green1, upper_green1, lower_green2, upper_green2, None, None)
    elif color_name.startswith("azul"):
        lower_blue1, upper_blue1 = color_ranges["azul_oscuro"]
        lower_blue2, upper_blue2 = color_ranges["azul_claro"]
        return (lower_blue1, upper_blue1, lower_blue2, upper_blue2, None, None)
    elif color_name.startswith("rosa"):
        lower_pink1, upper_pink1 = color_ranges["rosa_alto"]
        lower_pink2, upper_pink2 = color_ranges["rosa"]
        lower_pink3, upper_pink3 = color_ranges["rosa_bajo"]
        return (lower_pink1, upper_pink1, lower_pink2, upper_pink2, lower_pink3,
upper_pink3)
    elif color_name in color_ranges:
        lower_range, upper_range = color_ranges[color_name]
        return (lower_range, upper_range, None, None, None, None)
    else:
        print(f"Color '{color_name}' no encontrado. Usando rango de color blanco.")
        return ([0, 0, 200], [180, 30, 255], None, None, None, None)

def start_detection(color_name, video_path, status_label):
    # Cargar YOLO
    net = cv2.dnn.readNet("yolov4.weights", "yolov4.cfg")
    layer_names = net.getLayerNames()
    output_layers = [layer_names[i - 1] for i in net.getUnconnectedOutLayers()]
    classes = []
    with open("coco.names", "r") as f:
        classes = [line.strip() for line in f.readlines()]

    # Obtener los rangos de color
    lower_color1, upper_color1, lower_color2, upper_color2, lower_color3, upper_color3 =
get_color_ranges(color_name)

    cap = cv2.VideoCapture(video_path)

    fps = int(cap.get(cv2.CAP_PROP_FPS))
    post_detection_buffer_size = fps * 10
    buffer_size = fps * 5
```

```
frame_buffer = deque(maxlen=buffer_size)

output_video = None
fourcc = cv2.VideoWriter_fourcc(*'mp4v')
frame_size = None
saving_video = False
save_counter = 0

_, prev_frame = cap.read()
prev_frame_gray = cv2.cvtColor(prev_frame, cv2.COLOR_BGR2GRAY)

total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
processed_frames = 0

def stop_detection():
    nonlocal running
    running = False
# Asignar la función stop_detection al evento de presionar la tecla 'q'
root.bind('<q>', lambda event: stop_detection())

running = True
# Iniciar el bucle principal
while cap.isOpened() and running:
    ret, frame = cap.read()
    if not ret:
        break

    frame_original = frame.copy()
    frame_buffer.append(frame_original)

    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    frame_delta = cv2.absdiff(prev_frame_gray, gray_frame)
    thresh = cv2.threshold(frame_delta, 25, 255, cv2.THRESH_BINARY)[1]
    thresh = cv2.dilate(thresh, None, iterations=2)

    prev_frame_gray = gray_frame.copy()

    height, width, channels = frame.shape
    blob = cv2.dnn.blobFromImage(frame, 0.00392, (416, 416), (0, 0, 0), True, crop=False)
    net.setInput(blob)
    outs = net.forward(output_layers)

    class_ids, confidences, boxes = [], [], []
    for out in outs:
        for detection in out:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]
```

```
            if confidence > 0.5 and classes[class_id] == "person":
                center_x = int(detection[0] * width)
                center_y = int(detection[1] * height)
                w = int(detection[2] * width)
                h = int(detection[3] * height)
                x = int(center_x - w / 2)
                y = int(center_y - h / 2)

                boxes.append([x, y, w, h])
                confidences.append(float(confidence))
                class_ids.append(class_id)

        indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)
        color_detected = False

        if len(indexes) > 0:
            for i in indexes.flatten():
                x, y, w, h = boxes[i]
                person_roi = frame[y:y+h, x:x+w]

                person_movement = thresh[y:y+h, x:x+w]
                movement_detected = cv2.countNonZero(person_movement) > 0

                if movement_detected:
                    hsv_roi = cv2.cvtColor(person_roi, cv2.COLOR_BGR2HSV)

                    mask = cv2.inRange(hsv_roi, np.array(lower_color1), np.array(upper_color1))
                    if lower_color2 is not None:
                        mask2 = cv2.inRange(hsv_roi, np.array(lower_color2),
np.array(upper_color2))
                        mask = cv2.bitwise_or(mask, mask2)
                    if lower_color3 is not None:
                        mask3 = cv2.inRange(hsv_roi, np.array(lower_color3),
np.array(upper_color3))
                        mask = cv2.bitwise_or(mask, mask3)

                    mask_movement_color = cv2.bitwise_and(mask, person_movement)

                    total_moving_pixels = cv2.countNonZero(person_movement)
                    color_in_movement_pixels = cv2.countNonZero(mask_movement_color)

                    if total_moving_pixels > 0:
                        color_percentage = (color_in_movement_pixels / total_moving_pixels) * 100
                    else:
                        color_percentage = 0

                    contours, _ = cv2.findContours(mask_movement_color, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
```

```
                cv2.drawContours(frame[y:y+h, x:x+w], contours, -1, (255, 0, 0), 2)

                if color_percentage >= 10:
                    color = (0, 0, 255)
                    color_detected = True

                    if output_video is None:
                        folder_name = color_name.lower()
                        if not os.path.exists(folder_name):
                            os.makedirs(folder_name)
                        frame_size = (frame.shape[1], frame.shape[0])
                        timestamp = datetime.now().strftime('%Y%m%d_%H%M%S')
                        video_name = f"{timestamp}.mp4"
                        video_path_full = os.path.join(folder_name, video_name)
                        output_video = cv2.VideoWriter(video_path_full, fourcc, fps, frame_size)

                        cursor.execute(''' INSERT INTO videos (name, color, path)
                                        VALUES (?,?,?)''', (video_name, color_name, video_path_full))
                        conn.commit()

                        for buffer_frame in frame_buffer:
                            output_video.write(buffer_frame)
                        saving_video = True
                        save_counter = 0

                    if saving_video:
                        #frame_buffer.append(frame_original)
                        #resized_frame = cv2.resize(frame_original, frame_size)
                        frame_to_save = frame_original.copy()
                        cv2.rectangle(frame_to_save, (x,y), (x+w, y+h), (0, 0, 255), 2)
                        output_video.write(frame_to_save)
                        save_counter += 1
                        print(f"Frames guardados: {save_counter}")
                        if save_counter >= post_detection_buffer_size:
                            saving_video = False
                            output_video.release()
                            output_video = None
                            save_counter = 0
                            frame_buffer.clear()
                    else:
                        color = (0, 0, 0)
                else:
                    color = (0, 0, 0)

                cv2.rectangle(frame, (x, y), (x + w, y + h), color, 2)
                cv2.putText(frame, f"{classes[class_ids[i]]}: {confidences[i]:.2f}", (x, y + 30),
cv2.FONT_HERSHEY_SIMPLEX, 1, color, 2)
            # Si no se detectó color y se está grabando un video, guardar el frame en el buffer
```

```
            if not color_detected and saving_video:
                save_counter += 1
                #frame_buffer.append(frame_original)
                resized_frame = cv2.resize(frame_original, frame_size)
                output_video.write(resized_frame)
                print(f"Frames guardados (no color): {save_counter}")
                if save_counter >= post_detection_buffer_size:
                    print("Se detuvo la grabación porque ya no hay capacidad.")
                    saving_video = False
                    output_video.release()
                    output_video = None
                    save_counter = 0
                    frame_buffer.clear()

            cv2.imshow('Frame', frame)

            processed_frames += 1
            root.update_idletasks()

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

        cap.release()
        cv2.destroyAllWindows()

        if output_video is not None:
            output_video.release()

        status_label.config(text="Detección completada")

    # Función para mostrar la lista de videos guardados
    def show_saved_videos():
        # Crear una nueva ventana para mostrar la lista de videos
        video_window = tk.Toplevel(root)
        video_window.title("Videos Guardados")
        video_window.geometry("400x300")

        # Crear un Treeview para mostrar los videos
        tree = ttk.Treeview(video_window, columns=("ID", "Nombre", "Color", "Ruta"),
    show="headings")
        tree.heading("ID", text="ID")
        tree.heading("Nombre", text="Nombre")
        tree.heading("Color", text="Color")
        tree.heading("Ruta", text="Ruta")
        tree.pack(fill=tk.BOTH, expand=True)

        # Obtener los videos de la base de datos
        cursor.execute("SELECT * FROM videos")
```

```
    videos = cursor.fetchall()

    # Insertar los videos en el Treeview
    for video in videos:
        tree.insert("", tk.END, values=video)

    # Función para reproducir el video seleccionado
    def play_video(event):
        selected_item = tree.selection()[0]
        video_path = tree.item(selected_item, "values")[3]
        cap = cv2.VideoCapture(video_path)

        while cap.isOpened():
            ret, frame = cap.read()
            if not ret:
                break
            cv2.imshow('Video', frame)
            if cv2.waitKey(25) & 0xFF == ord('q'):
                break

        cap.release()
        cv2.destroyAllWindows()

    # Asignar la función play_video al evento de doble clic en el Treeview
    tree.bind("<Double-1>", play_video)

# Funciones para la interfaz gráfica
def select_video_file():
    file_path = filedialog.askopenfilename(filetypes=[("Video files", ".mp4;.avi")])
    if file_path:
        video_path.set(file_path)
        cap = cv2.VideoCapture(file_path)
        ret, frame = cap.read()
        if ret:
            # Pasar el frame a RGB y usar PIL para manejar la imagen
            frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            img = Image.fromarray(frame_rgb)
            preview_image = ImageTk.PhotoImage(img)
            preview_label.config(image=preview_image)
            preview_label.image = preview_image
        cap.release()

def start_detection_process():
    color_name = color_var.get()
    video_file = video_path.get()
    if not color_name or not video_file:
        messagebox.showwarning("Advertencia", "Seleccione un color y un archivo de video.")
        return
```

```
    status_label.config(text="Iniciando detección...")
    start_detection(color_name, video_file, status_label)

def reset_selections():
    color_var.set("")
    video_path.set("")
    status_label.config(text="")
    preview_label.config(image="")
    preview_label.image = None

# Crear la ventana principal
root = tk.Tk()
root.title("Detección de Movimiento y Color")
root.geometry("800x600")

# Variables
color_var = tk.StringVar()
video_path = tk.StringVar()

# Estilos
style = ttk.Style()
style.configure("TButton", font=("Helvetica", 12), padding=10, background="blue",
foreground="blue", borderwidth=0)
style.configure("TLabel", font=("Helvetica", 12), padding=10, background="#2c3e50",
foreground="#ecf0f1")
style.configure("TOptionMenu", font=("Helvetica", 12), padding=10, background="blue",
foreground="blue")

# Frames
header_frame = tk.Frame(root, bg="#34495e")
header_frame.place(relwidth=1, relheight=0.1)

sidebar_frame = tk.Frame(root, bg="#34495e", width=200)
sidebar_frame.place(relwidth=0.25, relheight=0.9, rely=0.1)

main_frame = tk.Frame(root, bg="#2c3e50")
main_frame.place(relwidth=0.75, relheight=0.9, relx=0.25, rely=0.1)

# Header
tk.Label(header_frame, text="Detección de Movimiento y Color", bg="#34495e",
fg="#ecf0f1", font=("Helvetica", 16)).pack(pady=10)

# Sidebar Widgets
tk.Label(sidebar_frame, text="Seleccione un color:", bg="#34495e",
fg="#ecf0f1").pack(pady=10)
color_menu = ttk.OptionMenu(sidebar_frame, color_var, "rojo", "naranja", "amarillo", "verde",
"azul", "violeta", "rosa", "blanco", "negro")
color_menu.pack(pady=10)
```

```
button_width = 20
button_height = 2

ttk.Button(sidebar_frame, text="Seleccionar video", command=select_video_file,
style="TButton", width=button_width).pack(pady=10)
tk.Label(sidebar_frame, textvariable=video_path, bg="#34495e", fg="#ecf0f1",
wraplength=180).pack(pady=10)

ttk.Button(sidebar_frame, text="Iniciar detección", command=start_detection_process,
style="TButton", width=button_width).pack(pady=10)
ttk.Button(sidebar_frame, text="Resetear", command=reset_selections, style="TButton",
width=button_width).pack(pady=10)

# Botón para mostrar los videos guardados
ttk.Button(sidebar_frame, text="Ver videos guardados", command=show_saved_videos,
style="TButton", width=button_width).pack(pady=10)

# Main Frame Widgets
preview_label = tk.Label(main_frame, bg="#2c3e50")
preview_label.pack(pady=20)

status_label = tk.Label(main_frame, text="", bg="#2c3e50", fg="#ecf0f1")
status_label.pack(pady=10)

# Iniciar el bucle principal de Tkinter
root.mainloop()

conn.close()
```

Aclaraciones

Es necesario tener en cuenta que no alcanza sólo con el código fuente para ejecutar el programa, sino que también es necesario tener los archivos correspondientes de **coco.names**, **yolovg.cfg**, y **yolov4.weights**. Los primeros dos están incluidos en el repositorio git. También es necesario tener distintos videos para usar de casos de estudio.

- yolov4.weights: https://drive.google.com/file/d/1QkmfxR-YJmHxqdQq32PYmkJY1HiIlVxD/view?usp=drive_link
- video usado para el caso de estudio: https://drive.google.com/file/d/1ZSBqH7PSspLX36_YwA5csXs4IdgIMG1f/view?usp=drive_link
- video usado para el color Amarillo: https://drive.google.com/file/d/14m9C7pokF29H0mba7KRK-O5B97ieoV3n/view?usp=drive_link
- video usado para el color verde: https://drive.google.com/file/d/1yaHirvM87IovqbIzFgwZj0aGzgJ01xQa/view?usp=drive_link