import cv2

import torch

import tempfile

from ultralytics import YOLO

import streamlit as st

from norfair import Detection, Tracker

import numpy as np

import torch.amp

# Load the YOLOv8 model for emergency vehicles

yolo\_v8\_emergency = YOLO('best\_emergency\_vehicle\_model.pt')  # Adjust path if needed

# Load the YOLOv5 model for non-emergency vehicles

yolo\_v5\_non\_emergency = torch.hub.load('ultralytics/yolov5', 'yolov5s')  # Pre-trained YOLOv5 model

# Define labels for emergency and non-emergency vehicles

emergency\_labels = ['Police Car', 'Police Van', 'Fire Truck', 'Ambulance']

non\_emergency\_labels = ['car', 'bus', 'truck', 'motorcycle']

st.title("ADVANCED TRAFFIC FLOW OPTIMIZATION FOR INTELLIGENT TRAFFIC SYSTEM - Emergancy Vehicle Detection")

# File uploader for multiple videos

uploaded\_files = st.file\_uploader(

    "Upload up to 4 Videos", type=["mp4", "mov", "avi", "mkv"], accept\_multiple\_files=True

)

def create\_detections(results, labels, model\_type="yolov8"):

    """Convert YOLO detection results to Norfair detections for tracking."""

    detections = []

    if model\_type == "yolov8":

        if isinstance(results, list):

            results = results[0]

        if hasattr(results, 'boxes'):

            for box in results.boxes:

                x1, y1, x2, y2 = map(int, box.xyxy[0])

                centroid = np.array([[(x1 + x2) / 2, (y1 + y2) / 2]])

                label = labels[int(box.cls)]

                conf = box.conf[0].item()

                # Check if the label matches any emergency or non-emergency vehicle

                if label in emergency\_labels + non\_emergency\_labels:

                    detections.append(

                        Detection(

                            centroid,

                            data={"label": label, "conf": conf, "box": (x1, y1, x2, y2)}

                        )

                    )

    elif model\_type == "yolov5":

        if hasattr(results, 'xyxy'):

            for result in results.xyxy[0]:

                if len(result) >= 6:

                    x1, y1, x2, y2, conf, cls = result[:6]

                    label = labels[int(cls)]

                    centroid = np.array([[(x1 + x2) / 2, (y1 + y2) / 2]])

                    if label in emergency\_labels + non\_emergency\_labels:

                        detections.append(

                            Detection(

                                centroid,

                                data={"label": label, "conf": conf, "box": (int(x1), int(y1), int(x2), int(y2))}

                            )

                        )

    return detections

if uploaded\_files:

    for idx, uploaded\_file in enumerate(uploaded\_files[:4]):  # Process up to 4 videos

        st.write(f"### Processing Video {idx + 1}: {uploaded\_file.name}")

        tfile = tempfile.NamedTemporaryFile(delete=False)

        tfile.write(uploaded\_file.read())

        cap = cv2.VideoCapture(tfile.name)

        stframe = st.empty()

        emergency\_detected = False  # Reset for each video

        # Unique IDs for each video

        unique\_emergency\_ids = set()

        unique\_non\_emergency\_ids = set()

        # Initialize ByteTrack tracker

        tracker = Tracker(distance\_function="euclidean", distance\_threshold=30)

        # Process each frame of the video

        while cap.isOpened():

            ret, frame = cap.read()

            if not ret:

                break

            # Run YOLOv8 model for emergency vehicle detection

            emergency\_results = yolo\_v8\_emergency(frame)

            # Run YOLOv5 model for non-emergency vehicle detection

            non\_emergency\_results = yolo\_v5\_non\_emergency(frame)

            # Generate detections from both models

            detections = create\_detections(emergency\_results, yolo\_v8\_emergency.names, model\_type="yolov8") + \

                         create\_detections(non\_emergency\_results, yolo\_v5\_non\_emergency.names, model\_type="yolov5")

            # Update tracked objects

            tracked\_objects = tracker.update(detections)

            # Draw bounding boxes on the frame

            for obj in tracked\_objects:

                label = obj.last\_detection.data["label"]

                x1, y1, x2, y2 = obj.last\_detection.data["box"]

                # If it’s an emergency vehicle

                if label in emergency\_labels:

                    if obj.id not in unique\_emergency\_ids:

                        unique\_emergency\_ids.add(obj.id)

                        emergency\_detected = True  # Trigger notification if emergency detected

                    cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 255), 2)

                    cv2.putText(frame, f'{label} {obj.id}', (x1, y1 - 10),

                                cv2.FONT\_HERSHEY\_SIMPLEX, 0.9, (0, 0, 255), 2)

                # If it’s a non-emergency vehicle

                elif label in non\_emergency\_labels:

                    if obj.id not in unique\_non\_emergency\_ids:

                        unique\_non\_emergency\_ids.add(obj.id)

                    cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)

                    cv2.putText(frame, f'{label} {obj.id}', (x1, y1 - 10),

                                cv2.FONT\_HERSHEY\_SIMPLEX, 0.9, (0, 255, 0), 2)

            # Update the frame in Streamlit

            # stframe.image(frame, channels="BGR", use\_container\_width=True)

            if frame is not None and isinstance(frame, np.ndarray):

                frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)  # Convert BGR to RGB

                stframe.image(frame, channels="RGB", use\_container\_width=True)

            else:

                st.warning("Frame is None or has an invalid format.")

        cap.release()

        # After the video ends, display the results

        if emergency\_detected:

            st.warning(f"🚨 Emergency vehicle detected in Video {idx + 1}. Please clear the road!")

        # Calculate and display road clearance time for this video

        non\_emergency\_count = len(unique\_non\_emergency\_ids)

        emergency\_count = len(unique\_emergency\_ids)

        clearance\_time = max(0, (non\_emergency\_count - emergency\_count) \* 3)

        st.write(f"### Results for Video {idx + 1}")

        st.write(f"Final Non-Emergency Vehicles: {non\_emergency\_count}")

        st.write(f"Estimated Road Clearance Time: {clearance\_time} seconds")