!pip install ultralytics filterpy

import cv2

import torch

import numpy as np

from ultralytics import YOLO

from sort import Sort # Install SORT manually if not included

# Load YOLOv8 Nano model (fast & lightweight)

model = YOLO("yolov8n.pt")

# Initialize SORT Tracker

tracker = Sort()

# Open Video File

video\_path = "input\_video.mp4" # Change to your video path

cap = cv2.VideoCapture(video\_path)

# Get video properties

frame\_width = int(cap.get(3))

frame\_height = int(cap.get(4))

fps = int(cap.get(cv2.CAP\_PROP\_FPS))

# Define Video Writer

output\_video = "output\_video.avi"

fourcc = cv2.VideoWriter\_fourcc(\*'XVID')

out = cv2.VideoWriter(output\_video, fourcc, fps, (frame\_width, frame\_height))

while cap.isOpened():

ret, frame = cap.read()

if not ret:

break

# Run YOLOv8 object detection

results = model(frame)

detections = []

for result in results:

for box in result.boxes:

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

conf = float(box.conf[0])

cls = int(box.cls[0])

# Confidence threshold (adjustable)

if conf > 0.3:

detections.append([x1, y1, x2, y2, conf])

# Convert to NumPy array

detections = np.array(detections)

# SORT Tracker Update

if len(detections) > 0:

tracked\_objects = tracker.update(detections)

else:

tracked\_objects = []

# Draw bounding boxes and IDs

for obj in tracked\_objects:

x1, y1, x2, y2, obj\_id = map(int, obj)

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

cv2.putText(frame, f'ID {obj\_id}', (x1, y1 - 10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 255, 0), 2)

# Write frame to output video

out.write(frame)

# Display (Optional: Comment out if running on Kaggle without GUI)

cv2.imshow("Tracking", frame)

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

break

# Release resources

cap.release()

out.release()

cv2.destroyAllWindows()

!pip install ultralytics filterpy

import cv2

import torch

import numpy as np

from ultralytics import YOLO

from sort import Sort # Ensure SORT is installed

# Load YOLOv8 Nano model (lightweight)

model = YOLO("yolov8n.pt")

# Initialize SORT Tracker

tracker = Sort()

# Input Video Path (Supports MP4 & AVI)

video\_path = "input\_video.mp4" # Change to your file

# Open Video File

cap = cv2.VideoCapture(video\_path)

if not cap.isOpened():

print("Error: Could not open video.")

exit()

# Get video properties

frame\_width = int(cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH))

frame\_height = int(cap.get(cv2.CAP\_PROP\_FRAME\_HEIGHT))

fps = int(cap.get(cv2.CAP\_PROP\_FPS))

# Output Video Path

output\_video = "output\_video.mp4" # Change to ".avi" if needed

# Select codec based on output format

if output\_video.endswith(".mp4"):

fourcc = cv2.VideoWriter\_fourcc(\*'mp4v') # MP4 format (H.264)

elif output\_video.endswith(".avi"):

fourcc = cv2.VideoWriter\_fourcc(\*'XVID') # AVI format

else:

print("Unsupported output format. Use .mp4 or .avi")

exit()

# Define Video Writer

out = cv2.VideoWriter(output\_video, fourcc, fps, (frame\_width, frame\_height))

while cap.isOpened():

ret, frame = cap.read()

if not ret:

break

# Run YOLOv8 object detection

results = model(frame)

detections = []

for result in results:

for box in result.boxes:

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

conf = float(box.conf[0])

cls = int(box.cls[0])

# Confidence threshold (adjustable)

if conf > 0.3:

detections.append([x1, y1, x2, y2, conf])

# Convert to NumPy array

detections = np.array(detections)

# SORT Tracker Update

if len(detections) > 0:

tracked\_objects = tracker.update(detections)

else:

tracked\_objects = []

# Draw bounding boxes and IDs

for obj in tracked\_objects:

x1, y1, x2, y2, obj\_id = map(int, obj)

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

cv2.putText(frame, f'ID {obj\_id}', (x1, y1 - 10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 255, 0), 2)

# Write frame to output video

out.write(frame)

# Display (Optional: Comment out if running on Kaggle without GUI)

cv2.imshow("Tracking", frame)

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

break

# Release resources

cap.release()

out.release()

cv2.destroyAllWindows()

print(f"Processed video saved as {output\_video}")