⊙ Object detection, tracking and counting system.

Required Tools:-

① Object detection - YOLO v8

② Tracking - Byte track

③ Counting - Supervision.

Use Case:- Counting cars on a street

NOTE:- ① Use GPU
② Use input video [CHANGE]

(3) Use Stable version of YOLO v8

(4) Get Byte track

(5) Install Roboflow's Supervision
↓
Be specific of 0.1.0 version via pip package.

(6) Load pretrained YOLOv8.pt model

(7) a new video can be generated → Train YOLOv8 from scratch on your custom dataset.

Tricky Part :-

(8) Use Supervision to recreate the inference loop

Q "How do I run the model on every frame of the video?"

↓

use YOLOv8 model i.e imported from python SDK to run inference inside that loop.

(8) Get a frame generator

Date: / /

↓

Generate frames from input video. (one by one)

(9) Optional: Show a single frame

→ Both the functions are available in Supervision library.

(10) Output of this will give numpy array of a frame.

(11) Optional: If you want to check whether Yolov8 detects vehicles in a single frame.

→ (with bounding boxes around the vehicles)

(12) optional

To check the confidence of these boxes use .conf

* Idea:- Will the vehicle 1 will talk vehicle 2.

Date: /

optional

(12) To check which bounding box maps to which class of yolov8, use .cls

(14) NOTE:- Supervision is a general purpose library

So, Convert what you will get from Yolo v8 to an object that is understandable by supervision.

detections

(15) This ~~Annotator~~ → BoxAnnotator class of supervision used to display bounding boxes on the frames.

(16) Annotate frame with these detections.

O/P Frame → vehicles
→ b boxes around vehicles
→ Confidence score.

Date: / /

(17) Optional: If we want to pass class id / label i.e:- car, truck etc. in place of confidence score of that bbox. Then pass labels. or take both class id and confidence.

or class name.

(classnames.dict:)

(18) uptill here, it is for a single frame.

Now, apply this method for the whole video.

(19) So, we need to loop through all frames coming from input video.

(20) Save all those processed frames in the form of the video.

→ This happens with a class named 'VideoSink' of supervision

(21) VideoSink needs two arguments

1. instance of video info class
2. path to output file

video resolution

fps

total frame count

(22) Get this video information first, only then use videoSink.

Import video info from supervision.

(23) Optional :- Use tqdm, i.e a python package that is used for checking the progress of processing.

* OBJECT DETECTION IN VIDEO DONE HERE. *

GOING FOR TRACKING THE VEHICLES

24 Byte tracker

Create an instance of bytetracker before main for loop,

[SIDE NOTE: a code block inserted to solve the calibration issue b/w YOLOV5 and Bytetracker.]

Use that instance inside the for loop to get the tracker id's.

25 Update Labels alongwith trackerid

26 Passes the video

* TRACKING DONE HERE *

GOING FOR COUNTING

26 MAKE A LINE

27 COUNT the objects crossing that line.

Date: / /

26.1 THINK ABOUT THE LOCATION of THE LINE.

Line_Start Line_End

26.2 Use a class called as 'POINT' from supervision library

26.3 These two points used in a constructor called as line_counter.

26.4 display that line → another annotater required.

27.1 We need to let the line know where are the detections so that it can determine whether or not it got crossed by them and

27.2 Annotate the line_counter at the inside end of for loop.

So it can be aware about current counter values.