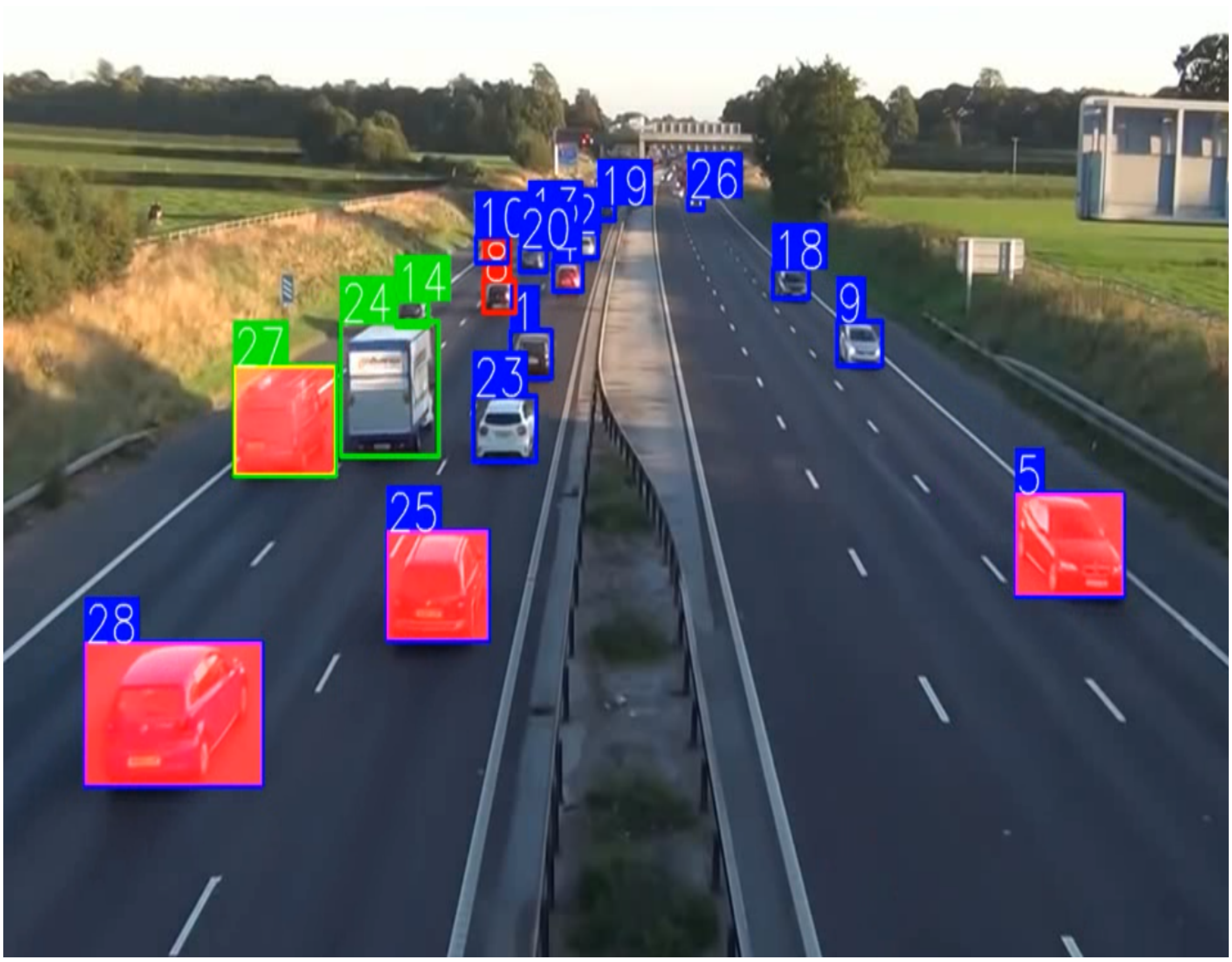
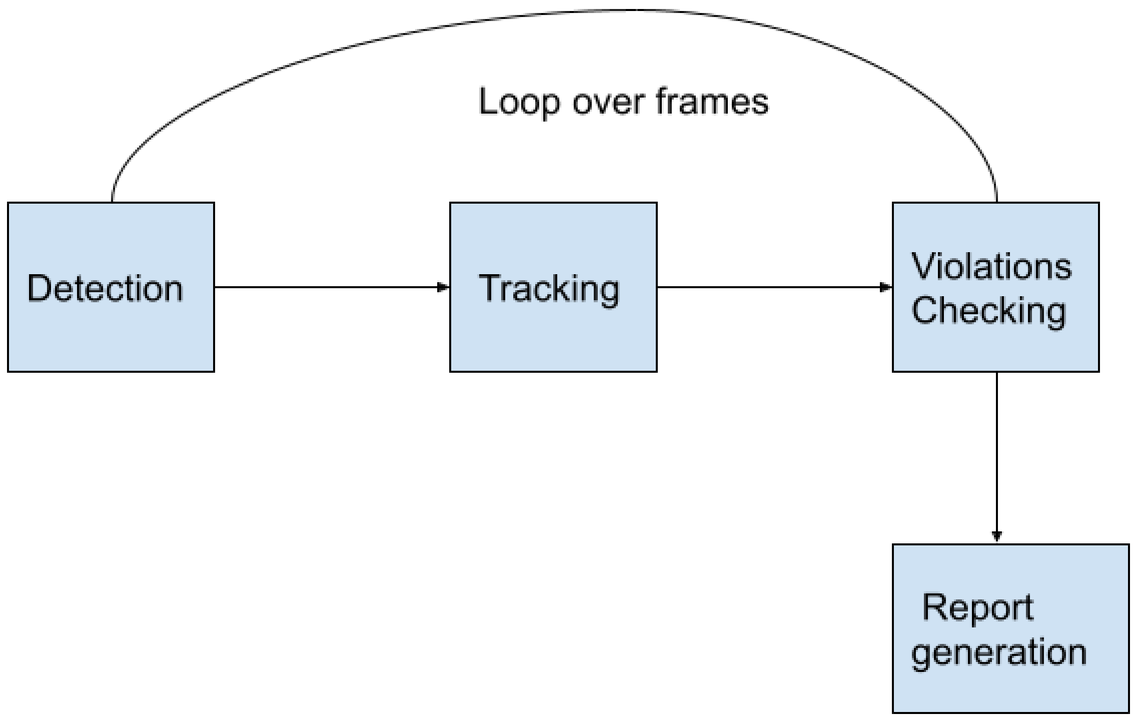


Street Rules Report

By Khaled Osama

Pipeline



1 - starts with car detection using yolov5s for getting bounding boxes over cars and defining their types (Car, Van, Truck).

2- passing these bounding boxes to the tracker so, the tracker is able to connect these boxes over time (by assigning to each boxe a unique ID).

3- after connecting boxes by the tracker and every car has its unique ID we can now compare the box of each car that appears in the current frame with the same object but with delta_frame before then calculate its speed by pixels/sec.

4- repeat from (1) to (3) over all video frames.

Modules

Detection Module

This module I've used Yolov5s with pre-trained on COCO dataset and train it over a VisDrone Datasets on only the 3 classes(Car, Van, Truck).

Dataset:

VisDrone Dataset which contains 10 classes. I filtered it and removed unneeded classes and kept only the 3 wanted classes (Car, Van, Truck). Then train it using a pretrained model on COCO dataset.

HyperParameters:

Evaluations:

Accuracy over testset

precision	Recall	map@0.5	F1 score
0.66	0.45	0.51	0.54

Tracking Module

This module I've used a deep sort tracking algorithm which is based on Kalman filter and siamese network.

HyperParameters:

I have used the default parameters with a few changes.
I have changed the number of frames that the box should appear in consecutive frames to be confirmed as track from 3 to 15 and change the maximum age from 40 to 20 which

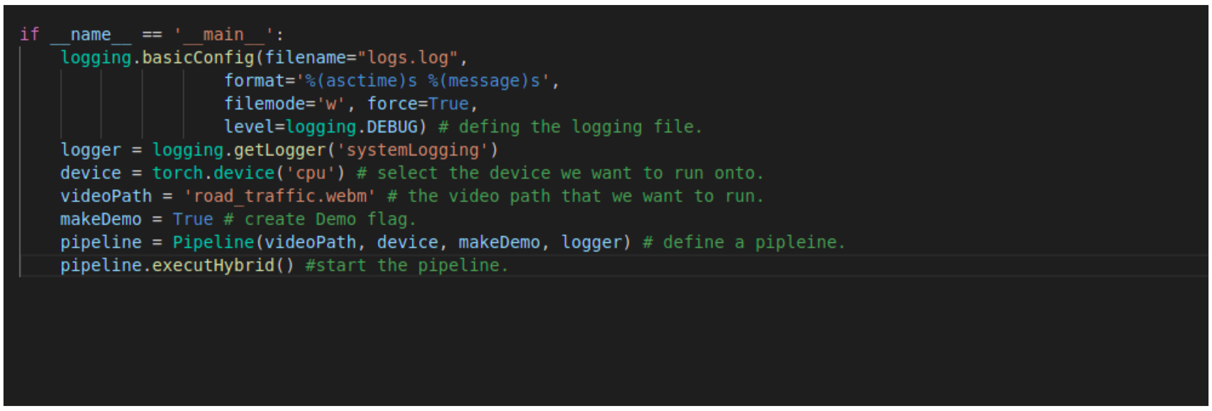
means that if the frame doesn't appear in 20 consecutive frames the tracker can consider that it disappears.

Violation Checking:

Every frame after the detector extracts the cars' bounding boxes and the tracker assigning Ids to each car this module can get the current frame boxes and get the same object from **current_frame_index - frame_delta** then compare every car bounding box in the current frame with the same car in the frame_index - frame_delta and calculate the distance between the 2 boxes centres by the euclidean distance so, we can get distance (**dst**) (which represents that the car makes a distance dst in a frame_delta time). So we can normalise to be pixel/second.

How to Run The system

There's a file requirements.txt which contains all the package and installation needed to run the system. There is a python file main.py which acts as an entry point to the system.



```python
if __name__ == '__main__':
    logging.basicConfig(filename="logs.log",
                        format='%(asctime)s %(message)s',
                        filemode='w', force=True,
                        level=logging.DEBUG) # defing the logging file.
    logger = logging.getLogger('systemLogging')
    device = torch.device('cpu') # select the device we want to run onto.
    videoPath = 'road_traffic.webm' # the video path that we want to run.
    makeDemo = True # create Demo flag.
    pipeline = Pipeline(videoPath, device, makeDemo, logger) # define a pipleine.
    pipeline.executHybrid() #start the pipeline.
```

Expected Output

- Violation_report.csv: which contains all violations in every frame, so every car can appear several times in different frames but this file can take further steps to get exactly what we want.
- video_Demo.avi: if the makeDemo flag is passed as True so, the video will be generated while the query video is processed.
- logs.log : the run logging file which tells us more details about this run.

Demo description

The demo will show all the results of the system. All vehicles should be bounded by boxes and every car type (Car, Van, Truck) should have different colours.

- Blue for Car
- Red for Van
- Green for Truck

Every vehicle should have an identifier which is a number displayed above the object box which shouldn't be changed for a specific vehicle.

If any vehicle speed exceeds the limit speed which in our case (Car => 120 pixel/sec, Van => 80 pixel/sec, Truck => 60 pixel/sec) so a red segment will fill the bounding box of that vehicle.

There are 2 counters in the top left of each frame that displays the frame number and

System Time Performance

The tracker time heavily depends on the number of boxes that it is trying to associate so, actually there's no fixed time. But on average on my machine core i7 it runs 8 frames per second in normal cases (sometimes it's faster and sometimes slower depending on how many vehicles are in the scene).

Notes

- The system shouldn’t detect any other vehicles rather than (car, Van , Truck)
- I added a mask which is in tracking/mask.jpg to neither detect or track the very far vehicles we can work without.

System Drawbacks

1- The dataset I have used has a huge imbalance of data so, we need to increase the minority classes (Trucks and vans) so, the system classifies the cars very well but can’t classify the trucks and vans very well.

2- The pixel/sec metric isn’t a good idea because if the vehicle takes steps near to the camera the rate of change of pixels over time for this vehicle is very high. On the other side when the vehicle is far away from the camera the rate of change will be low.

3- The detector needs more training for better results. I have stopped the training because of the time.