Summary of the car tracking task

This task has two main parts: 1) locating the vehicles, and 2) projecting the traffic on a custom image of the same road junction from another view angle.

Car localization
For the first task, the YOLOv5 model was used. YOLO is a family of object detection models trained on a large number of images (COCO dataset with ~200,000 labels). It has been found to be progressively successful in object detection tasks in terms of accuracy and speed. It is also a popular technique in the related community with comprehensive online documentation. This motivated to use YOLOv5 in this task for car localization.

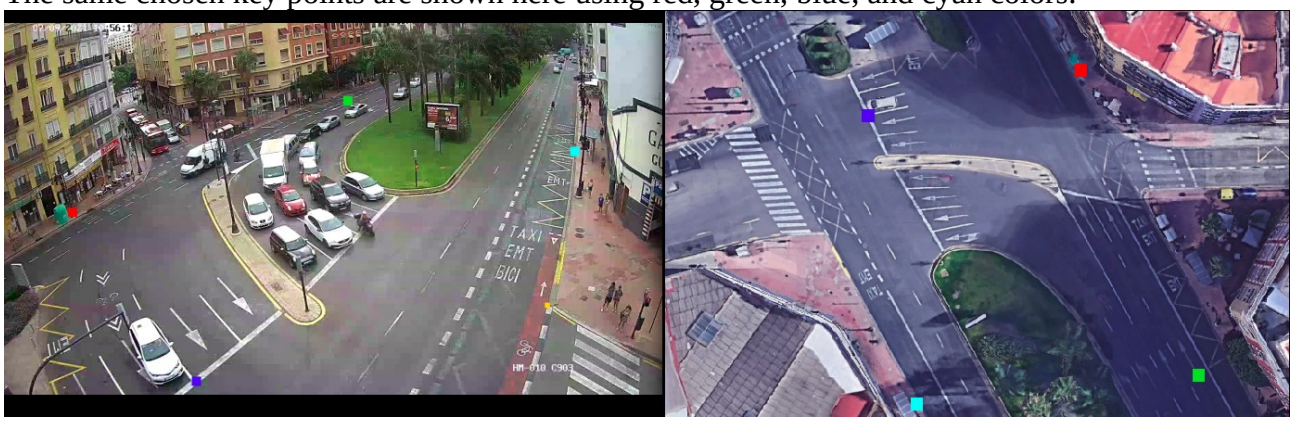
Some of the source codes from the YOLO was downloaded and modified to be able to run inference on the task video.

There are 5 pre-trained models available which are a tradeoff inference speed and object detection accuracy. Depending on the resources and expectations, one of these pre-trained weights could be used for the inference task:

1. yolov5n.pt is the fastest but least accurate
2. yolov5s.pt is slower than 'n' but more accurate
3. yolov5m.pt is balance between speed and accuracy
4. yolov5l.pt is more accurate than 'm' but slower
5. yolov5x.pt is the most accurate model but slowest

View projection from video to another view
The second part of this task is to project the localized cars into a custom view of the same road junction (google-earth).
For this task, the homography function by OpenCV library was used. For this part, 4 common key points between the camera view and the google-earth views were selected to estimate the warping matrix:
https://docs.opencv.org/4.x/da/d54/group__imgproc__transform.html#gaf73673a7e8e18ec6963e3774e6a94b87

The same chosen key points are shown here using red, green, blue, and cyan colors:



Once the warping matrix is calculated, it was used to project the coordinates of the localized cars into the google-eath image coordinte using general.warp_point function.

Aside from the manual selection of the key points in the two views, the SIFT method was also tried to automated the featuer matching but it failed and only one key point was found correctly between the two images.

Implementation
This repository is mostly based on YOLOv5 github account (https://github.com/ultralytics/yolov5). The implementation of the codes for this task are in the following python files: track_cars.py, utils.general, utils.plot, and warp_matrix_estimation.py.

Possible improvements
The YOLO pretrained weights could be potentially furthurer trained (fined tuned) on some labeled frames obtained from the same road junction.

The warping from camera view to google-earth view does not cover the whole google-earth view and it appears there is no car at the top of the screem. One may use a regression approach to forcast the trajectory of the cars towards the end of the camera FoW. This would increase the computational cost and does not seem to be necessary.

The feature matching for warping of the camera view to the google-earth view was manual. There are potential model-based solutions that could make this task automated.

The repository for this task is based YOLOv5 official github account and still has redundant python codes which are not needed for this task.