Video based vehicle detection, tracking, velocity estimation and registration number detection

Jiahao XU

Sibley School of Mechanical and Aerospace Engineering

Cornell University

MAE6760 Model Based Estimation

2021.12

Abstract:

During the courses of MAE6760: Model Based Estimation, a lot of filters originating from Bayesian Filters are concretely discussed. Hence, combining the course contents and the interests about the Autonomous Vehicles (AV), the final project is about using the edge-cutting computer vision techniques and the Kalman Filter to do the Video based vehicle detection, tracking, velocity estimation and registration number detection.

Backgrounds:

Nowadays, there are a lot of computer vision algorithms being used. The most famous ones are the YOLO and Faster RCNN. However, these algorithms are based on the images, so when it comes to videos, they cannot capture the relationships between each frame. And Kalman Filter is a good estimation tool for finding the connections of each frame and thus it will greatly improve the performance and even predict the hidden states.

Plan of Actions:

The outline of this project will include theses several key steps:

1. Using YOLOv5 to detect vehicles of each frame
2. Using Kalman Filter to match objects in each frame to achieve tracking (DeepSORT algorithm)
3. Vehicle counting based on its category
4. Velocity estimation
5. Registration number Recognition of each vehicle

YOLO algorithm for object detection

YOLO is currently the most widely used image target detection algorithm. As the first version of the YOLO series, YOLOv1 pioneered a single-stage algorithm. Compared with two-stage algorithms like RCNN series, it greatly shortened the prediction time and made real-time online prediction possible. At the same time, because it is a single-stage model, the detection of small targets is not very good, so subsequent models mainly improve it in this respect.

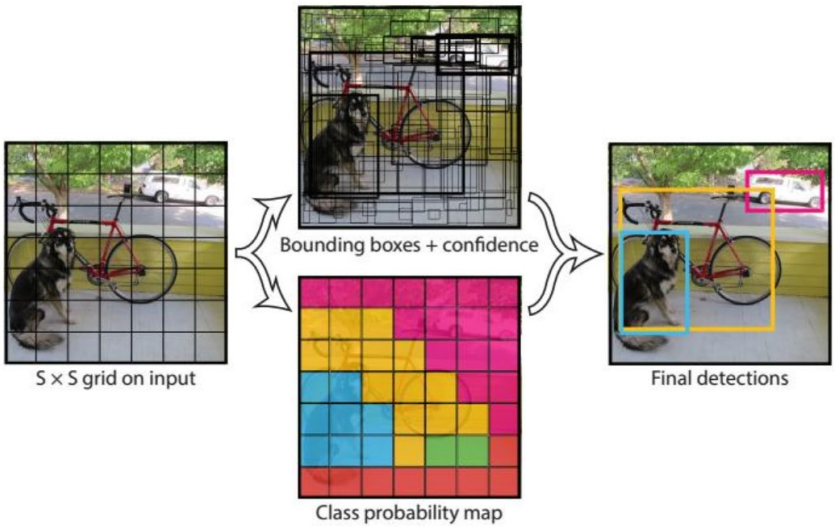

Fig1. YOLO v1 Object Detection Flow Chart

In the YOLOv1 algorithm, the object detection problem is treated as a regression problem. A convolutional neural network structure can directly predict the bounding box and category probability from the input image. The algorithm first divides the input image into S×S grids, and then predicts B bounding boxes for each grid, and each bounding box contains 5 predicted values: x, y, w, h and confidence. x, y are the center coordinates of the bounding box and they are aligned with the grid cell (the offset value of the current grid cell), so that their range is from 0 to

1. w and h are Normalized width and height (divide by the w and h of the image, so that the final w and h are in the range of 0 to 1). Confidence represents the confidence that the predicted box contains the object and the accuracy of this box prediction.

YOLOv1 has a poor detection result on objects that are very close to each other (the case where they are next to each other and their midpoints fall on the same grid). This is because only two boxes can be predicted in one grid, and they only belong to one type.

YOLOv5 is divided into four parts: Input, Backbone, Neck, and Prediction.

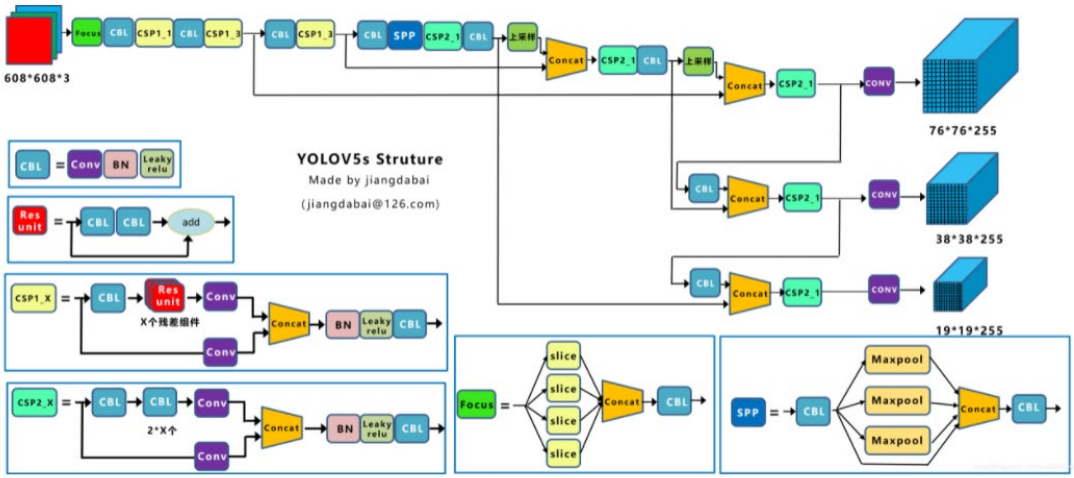

Fig2. YOLOv5 Structure

The specific calculation process of YOLOv5 is very complicated. This project mainly uses YOLOv5 for target detection and does not improve it, so it can be processed as a black box.

Kalman filter Based Tracking Algorithms

Object detection algorithms such as YOLO are based on images. For the problem of video-based object tracking, each frame has a certain continuous relationship, and we can regard the previous frame as the prior of the next frame. In this way, the tracking of the object can be realized using Kalman filter, and the classification of the object can be judged more accurately. The most widely used algorithms are SORT and DeepSORT.

SORT algorithm

The predecessor of DeepSORT is the SORT algorithm, and the core of the SORT algorithm is the Kalman filter and the Hungarian algorithm.

Kalman filter: the detection result of the first frame is used to initialize the state variable of the Kalman filter. Then use a series of states of the current frame to predict the states of the next frame.

1.The setting of SORT for state x is a seven-dimensional vector: x, y coordinates represent the position of the target center, s represents the relative size of the original target frame, r represents the aspect ratio relative to the original target frame, and the last three are horizontal, vertical, and scale changing rate, which are initialized to 0.

$$x = [u, v, s, r, \dot{u}, \dot{v}, \dot{s}]^T$$

2. For the Detection based Tracking framework, the next frame can be used as the detection result, so when using the Kalman filter, the detection result plays the role of the observation.

3. The setting of each covariance parameter uses empirical parameters. For each covariance of Kalman filter, the initial state is:

$$P = diag\left([10 \quad 10 \quad 10 \quad 10 \quad 1e4 \quad 1e4 \quad 1e4]^T\right)$$
$$Q = diag\left([1 \quad 1 \quad 1 \quad 1 \quad 0.01 \quad 0.01 \quad 1e{-}4]^T\right)$$
$$R = diag\left([1 \quad 1 \quad 10 \quad 10]^T\right)$$

It can be seen that the uncertainty of speed is greater than the uncertainty of position.

4. The next step is to determine the motion form and conversion matrix. Both SORT and DeepSORT are based on the assumption of uniform motion:

$$\begin{bmatrix} u \\ v \\ s \end{bmatrix} = \begin{bmatrix} u \\ v \\ s \end{bmatrix} + \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{s} \end{bmatrix} \Rightarrow F = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$z = Hx \Rightarrow \begin{bmatrix} u \\ v \\ s \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ s \\ r \\ \dot{u} \\ \dot{v} \\ \dot{s} \end{bmatrix}$$

5. For the update step of the Kalman filter, how to choose the observations is also a very important part, that is, for n tracking trajectories, if there are m observations, then there will be an allocation problem. SORT uses the Hungarian algorithm to first correlate the motion estimates and observations obtained in the Kalman filter prediction stage, and the cost matrix is the IOU between the two. If the matching is successful, the Kalman filter is updated. The tracking trajectory that fails to match is regarded as lost, and the observation that fails to match is regarded as a new trajectory.

The workflow of SORT is shown in the figure below and the Detections are the bounding boxes detected by the video and Tracks are tracking information.

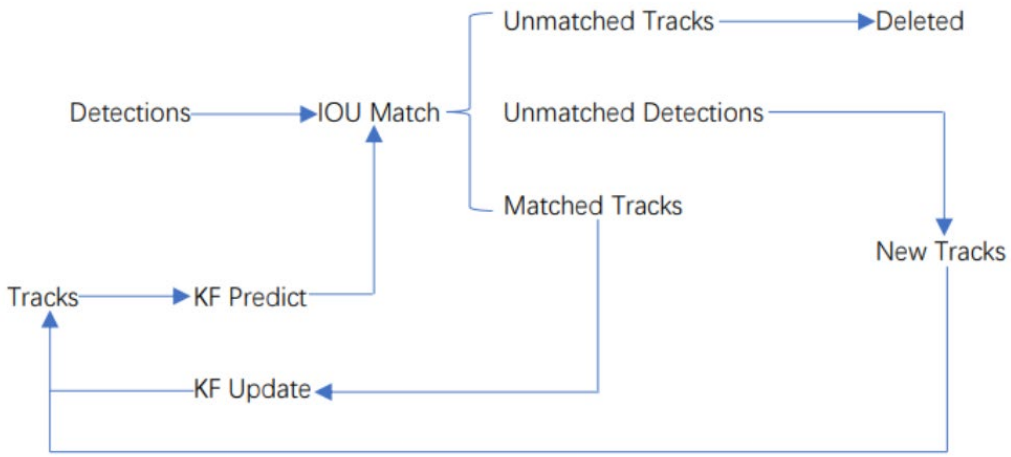

Fig3. The workflow of SORT algorithm

1. Create the corresponding Tracks from the results detected in the first frame. Initialize the states of the Kalman filter, and predict the bounding box of the next frame through the Kalman filter.

2. The bounding box obtained by YOLO of the next frame and the bounding box predicted by Tracks in the previous frame are matched with IOU one by one, and then the cost matrix is calculated based on the result of the IOU matching.

3. All the cost matrices obtained in step 2 are used as the input of the Hungarian algorithm, and then the matching result is obtained. There are three kinds of results we get at this time: 1.Tracks are mismatched (Unmatched Tracks), we directly delete them; 2. Detections are mismatched (Unmatched Detections), this usually means new objects enter FOV , so we initialize such Detections as a new Tracks; 3. Successful pairing of the detection frame and the predicted frame, which means that we have successfully tracked the target in the previous frame and the next frame, so we

pass the corresponding Detections through Kalman filter and updates its corresponding Tracks.

4. Repeat step 2 and step 3 until the end of the video.

DeepSORT algorithm

The SORT algorithm is still a rough tracking algorithm, because it can only find the relationship between only two frames. When the object is occluded, it is especially easy to lose its tracking ID. The DeepSORT algorithm is improved on the basis of the SORT algorithm to establish the long-term relationship between multi-frames and the method is to use the matching cascade and the confirmation of the new trajectory. Tracks are divided into confirmed state and unconfirmed state. Newly generated Tracks are in unconfirmed state; Tracks in unconfirmed state must be continuously matched with Detections for a certain number of times (default is 3) before they can be converted into confirmed state. Tracks in the confirmed state must be mismatched with Detections for a certain number of times (default 30 times) before they will be deleted.

Kalman filter in DeepSORT:

1. The state of the DeepSORT is an eight-dimensional vector. They respectively represent the coordinate of the target center, the aspect ratio and height of the current target frame, and the changing rate of the above four states.

$$x = \left[u, v, \gamma, h, \dot{u}, \dot{v}, \dot{\gamma}, \dot{h}\right]^T$$

2. For the Detection based Tracking framework, the next frame can be used as the detection result, so when using the Kalman filter, the detection result plays the role of the observation.

3. The setting of each covariance parameter should use empirical parameters. For each covariance of Kalman filter, the initial state is:

$$\begin{cases} P = diag\left(\left[2\sigma_p h \quad 2\sigma_p h \quad 1e-2 \quad 2\sigma_p h \quad 10\sigma_v h \quad 10\sigma_v h \quad 1e-5 \quad 10\sigma_v h\right]^T\right)^2 \\ Q = diag\left(\left[\sigma_p h \quad \sigma_p h \quad 1e-2 \quad \sigma_p h \quad \sigma_v h \quad \sigma_v h \quad 1e-5 \quad \sigma_v h\right]^T\right)^2 \\ R = diag\left(\left[\sigma_p h \quad \sigma_p h \quad 1e-1 \quad \sigma_p h\right]^T\right)^2 \end{cases}$$

It can be seen that compared to the SORT algorithm, two additional parameters, the standard deviation of the position and the velocity, are introduced. From the overall design, the uncertainty of the velocity relative to the position and shape is still higher.

4. The next step is to determine the motion form and conversion matrix. Both SORT and DeepSORT are based on the assumption of uniform motion but DeepSORT adds the step length of the movement, namely:

$$F = \begin{bmatrix} 1 & 0 & 0 & 0 & dt & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & dt & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & dt & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & dt \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$z = Hx \Rightarrow \begin{bmatrix} u \\ v \\ \gamma \\ h \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ \gamma \\ h \\ \dot{u} \\ \dot{v} \\ \dot{\gamma} \\ \dot{h} \end{bmatrix}$$

5. For the matching between the prediction step and the observations, DeepSORT introduces more settings: IOU is

used as the preliminary method, and the associations that do not meet the requirements are deleted. After that, the combination of Mahalanobis distance and cosine distance are used. The Mahalanobis distance uses the system covariance in the update phase of the Kalman filter.

$$d^{(1)}(i,j) = (d_j - y_i)^T S_i^{-1} (d_j - y_i)$$

The workflow of SORT is shown in the figure below.

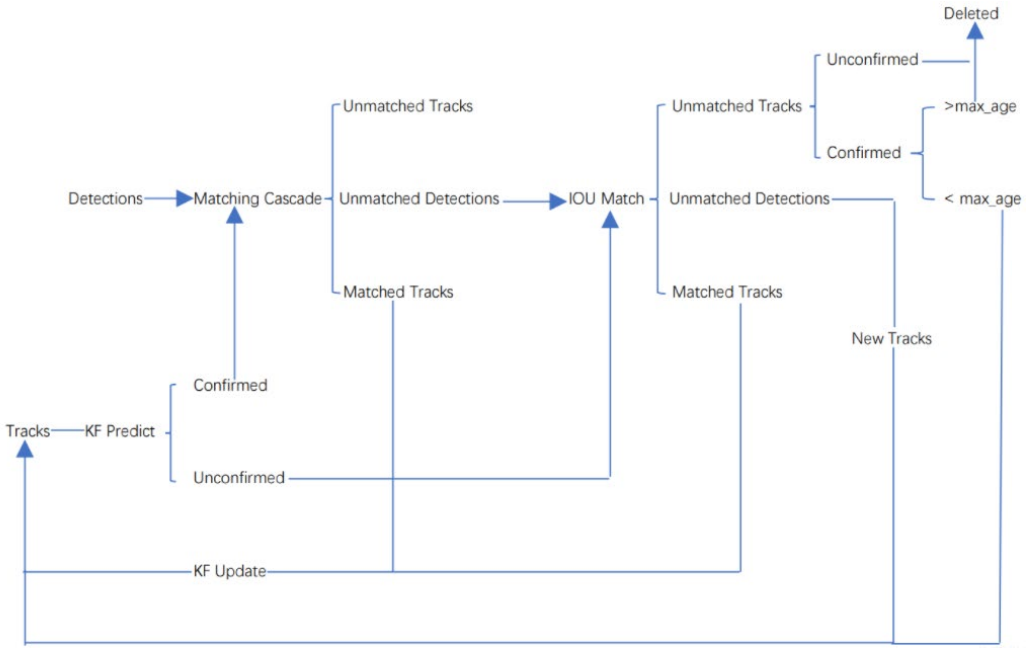

Fig4. The workflow of DeepSORT algorithm

1. Create the corresponding Tracks from the results detected in the first frame. Initialize the states of the Kalman filter, and predict its corresponding bounding box. The status of Tracks at this time is unconfirmed.

2. The bounding box detected by the target of this frame is matched with the bounding box predicted by Tracks in the previous frame one by one, and then the cost matrix is calculated based on the result of IOU matching.

3. Using the cost matrix obtained in step 2 as the input of the Hungarian algorithm, and then a linear matching result is obtained. There are three kinds of results at this time: 1. Unmatched Tracks, we directly delete the mismatched Tracks (because this Tracks is in an indeterminate state, if it is a certain state, it will have to reach a certain number of times in a row (default 30 times) before it can be deleted); 2. Unmatched Detections, we initialize such Detections as a new Track; 3. successful pairing of the detection frame and the predicted frame, which means that we have successfully tracked the previous frame and the next frame. The corresponding Detections are updated with the corresponding Tracks variable through the Kalman filter.

4. Repeat step 2 and step 3 until a confirmed track appears or the video frame ends.

5. Use Kalman filter to predict the bounding box of Tracks in the confirmed state and unconfirmed state. Cascade matching is used to match the bounding box of the confirmed Tracks and Detections(Because the confirmed Tracks and Detections are more likely to match).

6. There are three possible results after cascading matching. The first one is Tracks match, such Tracks update their corresponding Tracks variables through Kalman filter. The second and third is the mismatch of Detections and Tracks. At this time, the previous unconfirmed Tracks and the mismatched Tracks are matched with Unmatched Detections one by one for IOU matching, and then the cost matrix is calculated based on the result of the IOU matching.

7. Taking the cost matrix obtained in step 6 as the input of the Hungarian algorithm, and then a linear matching result is obtained. There are also 3 possible results like step 3.

8. Repeat from step 5 to step 7 until the end of the video.

Vehicle counting based on its category

This part is achieved by the information returned by the DeepSORT tracking algorithm: xywh, ID, and category. By setting a fixed detection line, when the bounding box passes the detection line, it is counted according to its category, and the ID of this box is recorded to prevent double counting, and the record of this ID is deleted when it leaves the detection area.

Velocity estimation

The most important part of video-based velocity estimation is the need to calibrate the actual distance. The pixel distance in the video needs to be converted to the actual distance. The velocity can be calculated based on the these information.

The main principle of real-time vehicle velocity estimation is to record the center pixel position of the ID at this moment after a fixed number of frames i, and use the Euclidean distance to calculate the pixel distance d1 moved during these frames. The vehicle width of the car is calibrated as d2=1.85m, so d2/w represents the actual distance represented by each pixel in the video at this time. (d2*d1)*fps/(w*i) is the formula of the final estimated velocity. And the camera has a certain angle difference, which needs to be corrected according to the actual vehicle velocity. There is no fixed formula here.

Registration number recognition

Recognition of the registration number is implemented using the third-party library HyperLPR. First, feed the bounding box of the detected vehicle to the model, and it will return the detected license plate number and confidence. The algorithm can update the license plate record based on the best one. For the same ID, when the confidence of the detected license plate is greater than the previous one, it will be replaced with the original license plate in the license plate record.

Final Deliverables

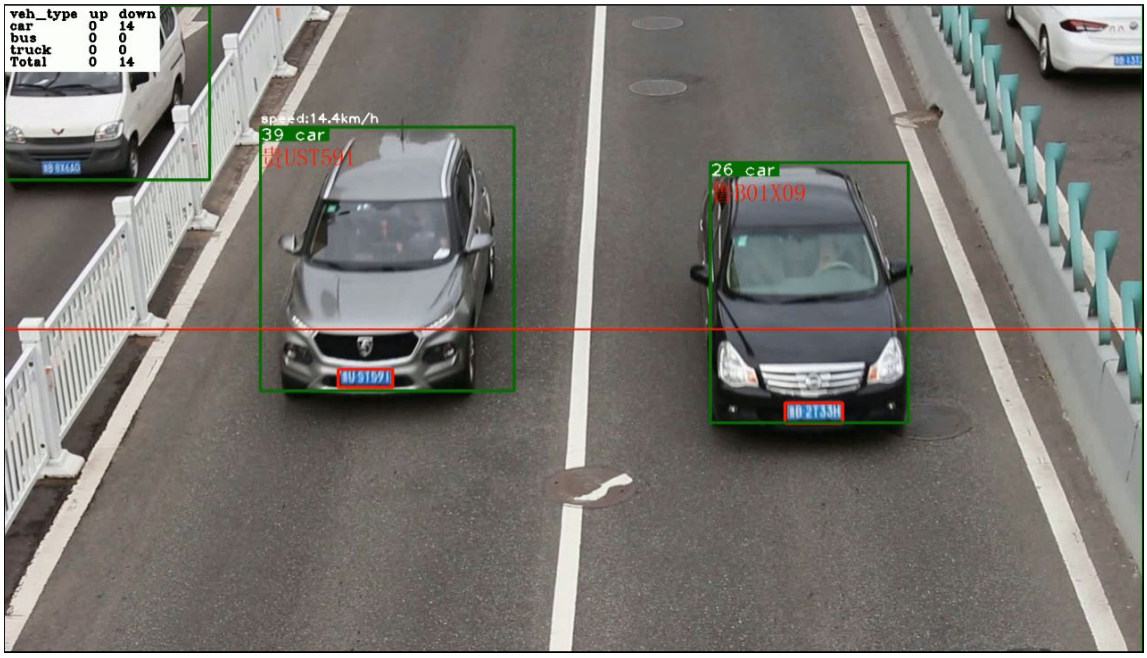

Fig5. Screenshot of the demo

Discussion/lessons learned/insights

From the demo video, it successfully achieved the vehicle detection of different categories; the bounding box is tracking the vehicles smoothly instead of popping up at every frame; and it also achieves a rough velocity estimation and registration number recognition.

During taking the course MAE6760, we studied a lot of filters and deeply compare their strength and potential weakness. Because these filters are originated from Bayesian Rules and Bayesian Rule is a great tool to find the relationship or connection for continuous systems. So, these filters are now become the baseline for a lot of Machine Learning and Computer Vision, especially for time series prediction and tracking. And also, one of the most charming parts is the mathematical simplification during using these filters. Because Bayesian Rule has three infinite integrals. It is hard to calculate in application. So, in the class we talk about 2 main kinds of simplification ways: Linearization (Extend Kalman Filter) and Monte Carlo Integration (Practical Filter). We can also take advantage of these 2 technical routines and apply them when facing complicated integration problems in our future research.

The algorithm of this vehicle speed detection is relatively rough, which is mainly limited by not knowing the pan, tilt and zoom of the camera, thus cannot perform FOV affine transformation. It can be calibrated later for improvement.

Appendix

The code is uploaded to GitHub:

https://github.com/JiahaoXU9/Video-based-vehicle-detection-tracking-velocity-estimation-and-registration-number-detection.git

git@github.com:JiahaoXU9/Video-based-vehicle-detection-tracking-velocity-estimation-and-registration-number-detection.git

The demo video is uploaded to YouTube:

https://www.youtube.com/watch?v=NK6a07Uq6L0