UNIVERSITY
OF TURKU

Multi-Object Tracking based Roadside Parking Behavior Recognition

Multi-Object Tracking
Master's Degree Programme in Information and Communication Technology
Department of Computing, Faculty of Technology
Master of Science in Technology Thesis

Author:
Hao Chen
Supervisors:
MSc (Tech). Xianjia Yu
Assoc. Prof. Tomi Westerlund

June 2022

Master of Science in Technology Thesis

Department of Computing, Faculty of Technology
University of Turku

Subject: Multi-Object Tracking
Programme: Master’s Degree Programme in Information and Communication Technology
Author: Hao Chen
Title: Multi-Object Tracking based Roadside Parking Behaviors Recognition
Number of pages: 59 pages
Date: June 2022

Abstract.

Roadside parking spaces can alleviate the shortage of parking spaces, but there are some shortcomings to the charges for roadside parking. The popular charging methods at present mainly include manual charging, geomagnetic detection charging, meter charging, etc. These methods have certain limitations, such as high cost, difficult deployment, and low acceptance of people. To solve the shortcomings of roadside parking charges, this thesis proposes a scheme based on deep learning and image recognition. More specifically, the thesis proposes a scheme for detecting and tracking vehicles, recognizing license plates, recognizing vehicle parking behavior, and recording vehicle parking periods through the monocular camera to solve the problem of roadside parking charges. The scheme has the advantages of convenient deployment, low labor cost, high efficiency, and high accuracy. The main work of this thesis is as follows:
1. Based on the You Only Look Once (YOLO) algorithm, this thesis proposes a trapezoidal convolution algorithm to detect objects and improve the detection efficiency for the problem that the vehicle is far and small in the image.
2. Proposes a one-stage license plate recognition scheme based on YOLO, aiming to simplify the license plate recognition process.
3. Depending on the characteristics of the vehicle, this thesis proposes a feature extraction model of the vehicle, called the horizontal and vertical separation model, which use to combine with the deep Simple Online and Real-time Tracking (SORT) object tracking framework to track the vehicle and improve the tracking efficiency.
4. Uses a Long Short-Term Memory (LSTM) model to classify the behavior of the vehicle into three types: Park, leave, and no behavior.
5. Groups these modules together, and the engineering code is debugged a lot to realize a complete Roadside Parking Behavior Recognition (RPBR) system.

Keywords: Roadside Parking; Vehicle Detection; Vehicle Tracking; Behavior Recognition.

Contents

1 Introduction

1.1 Background and significance

There are a large number of roadside parking spaces in the city. Due to the gradual increase in the number of vehicles in the city, roadside parking spaces are also extremely scarce. To efficiently use roadside parking spaces, car owners who park in these parking spaces are currently charged. Existing charging methods include manual charging, geomagnetic charging, meter charging, and so on. Although these charging methods are still improving, they have certain limitations after all.

At present, the advanced method of manual charging is to guard a certain area manually. When a vehicle is parked, the employee records the license plate through a proprietary recorder and prints out a receipt with a QR code. The billing time is recorded in the bill, which is used for billing. When the vehicle owner is about to leave, he scans the QR code of the receipt and pays to leave. Compared with the past, this method has indeed been greatly improved in efficiency, but it cannot break through the inherent limitations of this method. First, parking space is guarded manually, which requires a substantial labor cost; secondly, the problem of not discovering newly parked vehicles in time is prone to occur.

The geomagnetic induction charging mode uses geomagnetic sensors to sense the parking time of the vehicle to calculate the time it takes for the vehicle to park in the parking space. Recognize parked vehicles through geomagnetic recognition to prevent car owners from evading tolls. The main problems of this mode are high hardware cost and difficulty in installation and deployment. At the same time, it is difficult to popularize because this method requires components to be installed on the vehicle. The metered charging mode is to install a meter on the side of the parking space to detect whether the parking space has parked a vehicle. At the same time, its problems are similar to those of geomagnetic charging.

Compared with the previous methods, the method of monitoring roadside parking spaces through a monocular camera has many advantages. The first advantage is that the cost is relatively low. At present, the coverage rate of the cameras in the city is quite high. The use of installing cameras does not require any additional costs. At the same time, this method does not require manual supervision, which also reduces the cost. The second advantage is the easy deployment. Compared with geomagnetic induction and meter, the deployment of the camera is more convenient. The third advantage is a high generalization. Compared with geomagnetic induction and meter, the method of monitoring through a monocular camera does not need the vehicle owner to alter his beloved car, and all records can be processed in the server backend.

Why does this article not judge whether the vehicle has parking behavior directly by whether the position of the vehicle is in the parking space, but uses real-time tracking of the vehicle to determine whether the vehicle has parking behavior? Because of the actual situation, when multiple vehicles are parked in the parking area, there will be occlusion between the vehicles, so that the camera cannot obtain the license plate number of the vehicle. When the real-time tracking method is adopted, the camera can extract the license plate number in a picture where the license plate is not blocked. After the license plate number is extracted, it is not important whether the license plate number will be blocked.

1.2 Related works

1.2.1 Research status of domestic and foreign

YU Qing-qing and ZHANG Ke published a paper on Roadside Parking Detection and Recognition in 2019[1]. The paper concludes with KLT[2] and real-time compression tracking algorithm to extract the moving trajectory of the vehicle in the video, and extract the movement characteristics of the vehicle through the moving trajectory. Then use Support Vector Machine (SVM) to classify the motion characteristics to determine whether the vehicle has parking behavior[3]. In this paper, 25 videos are used for training, and 25 videos are used for testing; the video contains vehicles in 3 colors of black, white and gray, and 5 parking behaviors are classified at the same time. Got 95.60% accuracy. The article does not mention whether the data set comes from a certain fixed road section, and does not mention the recognition speed.

HU Xu, HUANG Jun also published a paper on Roadside Parking in 2019[4]. In this paper, a fast Gaussian Mixture Mode background modeling method is proposed, which separates the moving target from the background, and then uses the particle filter algorithm to track multiple objects to obtain the vehicle's trajectory. Then through the establishment of a mathematical model to classify the movement trajectory to determine whether the vehicle has parking behavior. The recognition accuracy and recognition speed are not given in the article.

In addition to the above two articles on parking recognition behavior detection, there are also a large number of papers on vehicle recognition and tracking. In addition, there are a large number of papers on dynamic target detection. If the dynamic target detection method is integrated into the target detection, the target detection speed can indeed be improved, but the dynamic target detection method is actually not suitable for tracking vehicles with parking behavior, because the speed of the vehicle is slow during the parking process. And it is very likely to stop. The reason for parking may be because the forward and backward changes, or it may be waiting for other moving objects to leave the trajectory that the vehicle is about to go.

1.2.2 Object detection and recognition algorithm

Vehicle detection and recognition rely on object detection and recognition algorithms. Due to the rise of deep learning, the current research on object detection and recognition has begun to tend to the research of deep learning networks. Object detection algorithm is a deep learning algorithm in the field of computer vision, if trained properly, can accurately identify objects in images or videos. This section gives an overall review of object detection. The first part starts with R-CNN[5] and introduces object detection based on Regional Proposal Network (RPN), including Fast R-CNN[6], Faster R-CNN[7] and R-FCN[8]. The second part focuses on one-stage detectors such as YOLO, SSD[9], Retina Net and so on, they are currently excellent detectors.

At present, object detection is divided into one-stage and two-stage. Two-stage means that the recognition process needs to be completed in two steps. The first step is to obtain candidate regions, and the second step is to classify the candidate regions, such as R-CNN. Series; in contrast, one-stage detection, its recognition process only needs one step, and does not have to find candidate regions separately, typically SSD and YOLO.

1.2.2.1 Two-stage: object detection based on e regions proposal

Two-stage is one of the basic deep learning object detection algorithms. The object detection process is mainly completed by a complete convolutional neural network, so the Convolutional Neural Network (CNN) feature will be used, and the description of the feature of the candidate area target is extracted through the convolutional neural network. Typical representatives: R-CNN to faster R-CNN. The two-stage algorithms can be regarded as end-to-end algorithms if the process of training the RPN network separately are not considered. But since they require two steps to get the result, they are not fully end-to-end algorithms. The first step of Two-stage is to train the RPN network to get the position of the object, and the second step is to classify. Compared with traditional detection algorithms, there is no need for additional training of classifiers, the process of feature representation, and the entire process of target detection is completed through a complete CNN from A to B. Compared with traditional algorithms, its accuracy is improved, while no additional classifiers need to be trained. Compared with the one-stage method, the accuracy is higher, but the speed is slower because it requires two steps to complete.

The basic process of the two-stage algorithm is to first input the image, and then perform deep feature extraction on the image, through the convolutional neural network, called the backbone network. And then complete the task completed by the sliding window in the traditional target detection algorithm through the RPN network. That is, the candidate area is generated. At the same time, the classification of the candidate frame is completed. This classification process will be divided into two categories of background and target. And the RPN network will make a preliminary prediction of the location of the target. Then a roi_pooling layer is needed to perform further accurate regression and correction of the candidate area. Next, after obtaining the regional feature of the candidate target corresponding to it on the feature map, a fully connected layer will be used to further express the features of the candidate region. Then through the two branches of classification and regression, the judgment of the candidate target category and the refinement of the position are completed respectively.

1.2.2.2 One-stage object Detection

Faster R-CNN uses a dedicated RPN to extract candidate regions. Region-based detectors are highly accurate, but inefficient. Faster R-CNN runs 7 frames per second on the PASCAL VOC[10] 2007 test set. But think about it, if our purpose is object detection, is it redundant to use a dedicated RPN network to extract candidate regions? Can't we get the bounding box and class of the object directly in one step?

In the sliding window detector, we can detect the object by sliding the window on the feature map. For different target types, we use different window types. The fatal mistake of the previous sliding window method is to use the window as the final result or bounding box, which requires a lot of shapes to cover most of the targets. A more effective method is to use the window as the initial window or initial bounding box, so that get a detector that predicts both the category and the bounding box together from the current sliding window.

One-stage algorithms such as YOLO and SSD, the main idea is that the convolution operation will not change the relative position of the object features, that is to say, when the object is in the upper right corner of the image, no matter how many convolutions, the features related to this object will only exist in the upper right corner of the convolution result. During the convolution process, both the features of the object and the position of the object are obtained. The features of the object are used for classification, and the value of the relative results of the object is the value of the bounding box. The whole process only needs one step. Therefore, the efficiency of the one-stage algorithm is very high, but it also has an important disadvantage that it is difficult to train, mainly because if the samples are extremely unbalanced, the accuracy of the model will be relatively low.

1.2.3 Object tracking algorithm

1.2.3.1 Classic object tracking algorithm

Tracking algorithms can be divided into generative models and discriminative models.

1. The generative model reflects the similarity of the same category. This type of method first establishes an object model or extracts object features, and searches for similar features in subsequent frames. Gradually iteratively achieve object positioning. Their disadvantage is that the accuracy of object tracking is not high in the case of illumination changes, motion blur, low resolution, and object rotation deformation. Common algorithms are Optical Flow Method[11], Particle Filter[12], Meanshift algorithm[13], Camshift algorithm[14], KCF[15], SRDCF[16], EBT[17].

2. The discriminative model of detection and tracking is to extract the object by comparing the difference between the object and the background information, and then get the position of the object in the current frame. Common algorithms are MIL[18], OAB[19], struck[20], MEEM[21], TLD[22], SVM.

The tracking algorithms can be divided into early tracking models, correlation filtering models and deep learning models.

1. Early tracking models are mostly generative models. The subdivision mainly includes three models. 1st, models based on object modeling, such as region matching,

feature point tracking, tracking algorithms based on object contours, optical flow methods, etc. 2nd, search-based method, in order to reduce the search range, a way to reduce the search range by prediction, such as Kalman filter, particle filter, etc. 3rd, kernel algorithm, this method continuously iterates the object template through gradient descent until the template reaches the optimum, such as Meanshift and Camshift.

2. Correlation filtering model: It is a discriminant model. Before correlation filtering, all tracking is performed in the time domain, involving complex matrix inversion calculations, and the speed is slow. Correlation filtering is performed in the frequency domain, and the circulant matrix can be diagonalized in the frequency domain, reducing the amount of calculation. Common algorithms are MOSSE[23], CSK[24], KCF, BACF[25], SAMF[26].

3. Deep learning model: Based on correlation filtering, deep learning is used to improve feature extraction and feature search.

1.2.3.2 SORT and Deep SORT tracking algorithms

The task of Multiple Object Tracking (MOT) is to track objects such as pedestrians, vehicles in a video or image sequence. In the process of tracking objects in video or in image sequence, even if the same object changes in position, aspect ratio, contrast, etc. in different frames, or is occluded for a short time, the object still has only a unique and constant ID, then Indicates that the tracking was successful.

The full name of SORT is Simple Online and Real-time Tracking. For the current multi-target tracking, it is more dependent on the quality of its detection performance, which means that by changing the detector, it can be improved by 18.9%. Although the SORT[27] algorithm in this article is just a combination of ordinary the algorithm such as Kalman Filter and Hungarian Algorithm combined together, but can match the SOTA algorithm 2016, and the speed can reach 260Hz, 20 times faster than the former.

In recent years, in the field of target detection algorithms, people have gradually started to use tracking-by-detection-based algorithms, which have gradually become the mainstream in MOT. Previous algorithms, such as flow network formulas and probabilistic graphical models, etc., is a global optimization problem dealing with the whole process, but due to the slow inference speed, it is not suitable for scenarios that require real-time recognition. Recently, some more traditional algorithms, such as Multiple Hypothesis Tracking[28] and Joint Probabilistic Data Correlation Filter[29], have been re-emphasized due to their success in object recognition. These methods are based on frame-by-frame data association for image recognition.

The SORT algorithm mainly combines the Kalman filter algorithm and the Hungarian algorithm to track objects. The computation of this algorithm is small, so it performs well. But its disadvantage is that it ignores the surface features of the detected object, which causes failure of tracking after objects being occluded. Deep SORT[30] is an upgraded version of SORT. It uses CNN feature extraction network to extract the features of objects. For example, if we need to track pedestrians, then we train the CNN feature extraction network for pedestrians. If we want to track vehicles, then we train about vehicles. CNN feature extraction network's purpose is to correctly extract the differences of each object. After adding the CNN feature extraction network, Compared with SORT, Deep SORT reduces about 45% of the ID switches.

1.2.4 Vehicle and license plate recognition algorithm

Silva[31] gave a complete license plate recognition system. The system focuses on solving the challenge of license plate recognition applications in non-restricted scenarios. Many databases commonly used in license plate recognition papers are often taken from the front, but in practical applications, there are various possible situations. But in his thesis, the author not only recognizes the front license plate, but also recognizes the license plate shot from the side, and finally achieved good results.

The license plate recognition system proposed by the author mainly has three steps: vehicle detection, license plate detection and correction, and Optical Character Recognition (OCR). The first is vehicle detection. The YOLOv2 without any modification is used in his thesis for vehicle detection; then the detected vehicle image is input to the Warped Planar Object Detection Network to perform the return of the license plate detection and the license plate curl correction system; finally, the license plate is corrected and input into OCR-Net network, recognize the license plate characters.

1.3 Research content and objectives

At present, the development of computer vision is quite rapid, especially in the direction of video classification, and the implementation ability is quite strong. Therefore, the research goal of this article is to apply computer vision-related technologies to roadside parking recognition to realize a real-time detection system for roadside parking behavior. The system can quickly and accurately identify roadside parking behaviors and vehicles driving off the roadside parking behaviors, and at the same time meets strong compatibility, and can be applied to various road sections, all time periods, regardless of day and night, light or dark. At the same time, the system can identify the license plate of the vehicle, which is used to record the parking behavior of a certain vehicle, so as to facilitate the collection of corresponding parking fee.

This thesis mainly researches the existing deep learning algorithms and frameworks, and then transforms the deep learning network according to actual scenarios, applies it to the scenarios required by this thesis, and achieves good results.

The research content of this thesis mainly includes four modules, which are vehicle detection and recognition, license plate recognition and binding, multi-object tracking and vehicle trajectory classification. This thesis not only researches the algorithms of each module, but also integrates each module to make it a complete roadside parking behavior detection system.

1.4 Thesis work and contributions

At the application level, the contribution of this thesis is to apply the popular computer vision technology to practice, making the installation and deployment process of vehicle parking behavior detection simpler and more advanced, which not only reduces labor costs but also makes roadside parking behavior recognition more accurate.

At the technical level, this thesis reforms the existing framework or algorithm to make it more suitable for roadside parking behavior recognition. More specifically. First, this thesis proposes a Trapezoidal Convolution Algorithm, which can greatly increase the inference speed while achieving a slight reduction in accuracy in the scenarios used in this paper. Then, this thesis proposes a one-step recognition algorithm based on the object recognition algorithm, which removes the cumbersome process in the traditional license plate recognition algorithm. Then, this thesis replaces the Reid model in the deep SORT framework, we proposed a feature extraction network model with horizontal and vertical separation. Compared with the previous model, the network model can improve the inference speed of the network model with a slight loss of accuracy. Last, this thesis proposes a real-time parallel trajectory classification algorithm based on LSTM[32], based on the memory characteristics of LSTM, realizes real-time trajectory classification of multiple vehicles.

1.5 Structure of thesis

This thesis is divided into four parts. Chapter 1 is the first part: an overview of this thesis. Chapter 2-5 is the second part, which introduces the technical details of the four modules of the RPBR system to be implemented in this thesis. Chapter 6 is the third part, which introduces the realization process of the RPBR system. Chapter 7 is the fourth part, summarizing the advantages and disadvantages of this thesis, as well as future work.

Chapter 1 is the introduction. This chapter first introduces the research background and significance of the paper and proposes the convenience of realizing parking behavior recognition for people's lives by using the currently popular deep learning and computer vision methods. Then in chapter 1.2, the research status of the technology used in parking behavior recognition from domestic and foreign is described. And also in chapter 1.2, we introduce the Object detection and recognition algorithms based on one-stage and two-stage, Object tracking algorithms based on the classic method and neural network, and Vehicle and license plate recognition algorithms. Subsequently, the research content and objectives of this paper are described, as well as the Thesis work and Contributions of this thesis. Finally, a brief introduction to the structure of this thesis is given.

Chapter 2 is about vehicle detection and recognition. First, it introduces the current research situation of object detection and recognition, and then introduces the application scenarios of this thesis and reasons for choosing YOLO as the basic algorithm. At the same time, this thesis proposes the trapezoidal convolution algorithm, which aims to improve the inferred speed of the model and introduces its working principle in detail. Finally, this chapter uses our data set to train and analyze the effect of the model that uses trapezoidal convolution and then compares it with the model that does not use trapezoidal convolution.

Chapter 3 is about license plate recognition and binding. Firstly, the current research status of license plate recognition is introduced. The current research on license plate recognition still focuses on positioning first and then recognition. Then this thesis proposes a one-step license plate recognition scheme through YOLOv5. Finally, it introduces how this thesis realizes the association between the license plate and vehicle through the program and solves some problems encountered when the vehicle is associated with the license plate.

Chapter 4 is about vehicle tracking. First, it roughly describes the current research status in the field of object tracking and then focuses on the related content of Deep SORT, a representative object tracking framework with good performance. After in-depth understanding and research on the Deep SORT framework and the Reid network model in it, this thesis proposes a network model HVS model suitable for the vehicle parking behavior recognition scene in this thesis. Finally, it briefly introduces the training process of the model and its comparison with the traditional Reid model.

Chapter 5 is vehicle trajectories classification. Firstly, it is introduced that the problem to be solved in this thesis is to realize the classification of trajectories, while also satisfying the real-time performance and the parallelism of multiple trajectory classification. Because of the uncertainty of the length of the vehicle trajectory, LSTM was finally selected as the basic algorithm for the trajectory classification solution. Then it introduces how to use the memory characteristics of LSTM to realize the classification of vehicle trajectories. Finally, the model is trained and the evaluation results are given.

Chapter 6 is implementation and result of the RPBR system. Firstly, this chapter gives an overview of the system. The system contains 4 modules, which are the Vehicle detection and recognition module introduced in detail in Chapter 2, the License Plate Recognition module introduced in Chapter 3, the Vehicle tracking module introduced in Chapter 4, and the Vehicle Trajectory Classification module introduced in Chapter 5. Then it introduces the integration process of each module in the system in detail. Then, this chapter explains the functions included in the system, explains the output format, or shows the output effects of the system. Then, this chapter tests the functionality of the system to see if there are serious systemic vulnerabilities. Finally, the system is summarized and conclusions are drawn.

Chapter 7 is Conclusion and future work. This chapter summarizes the work done in the full thesis and puts the outlook based on the problems that existed in the RPBR system of this thesis.

2 Vehicle detection and recognition

This chapter mainly introduces how to achieve vehicle detection and recognition, that is, how to mark all cars with bounding boxes in a video frame. Bounding box contains 4 values, each of them less than 1, which are the ratio of the abscissa of the bounding box to the width of the image x, the ratio of the ordinate of the bounding box to the height of the image y, the ratio of the width of the bounding box to the image width w, the ratio of the height of the bounding box to the height of the image h.

2.1 Basic algorithm selection

For the detection of vehicles in the video, this thesis combines the YOLO algorithm with the actual situation to improve some algorithms or steps in YOLO, so that it can achieve the same accuracy as YOLO in the scenes mentioned in this thesis, but the recognition faster.

At present, with the development of deep learning, the development of target detection and classification of images has made rapid progress. Many excellent algorithms have emerged in the field of object detection and classification. The first is the two-stage algorithm. The two-stage algorithm divides the steps into two steps: The first step is interest region extracted. There may be many objects of interest in an image, such as many different cars and shapes. Different pedestrians, etc., as well as areas that we are not interested in, such as the background. This step extracts the region of interest in the image; after extracting, it goes to the second step to classify the regions of interest, because we extract After finding the region of interest, it is not necessarily what we need. For example, there are cars, pedestrians, trash cans, etc. in the region of interest, but we are only interested in cars, so we need to classify them at this time. The more popular two stage algorithms are R-CNN, Fast R-CNN, Faster R-CNN, etc. Corresponding to the two-stage is the one-stage. As the name suggests, the one-stage algorithm only needs to be inferred once to get the position and category of all objects of interest in the image. YOLO is currently the more popular one stage algorithm, and it is also the algorithm we will focus on in this chapter.

Compared with the two-stage algorithm, the one-stage algorithm has both advantages and disadvantages. The advantage of the one-stage algorithm is that it is faster than the two-stage algorithm because it only needs to infer once to get the result. The disadvantage is that its accuracy is lower than the two-stage algorithm. The author of YOLO compared YOLO with Fast R-CNN, which worked well at the time[33]. He used the PASCAL-VOC dataset for comparison. First, he compared the calculation speed. The speed of the YOLO algorithm was better than other algorithms, the speed can reach 45 FPS, of which Fast YOLO can even reach 155 FPS, but the accuracy rate has dropped from 63.4 mAP to 52.7 mAP. Then compare the Fast R-CNN model with the highest mAP with YOLO. The mAP of Fast R-CNN reached 71.8%, while the mAP of YOLO was 63.4%. Although there is a gap, it is not particularly large. However, YOLO has an advantage over Fast R-CNN. The misjudgment rate of YOLO on the background (4.75%) is much lower than that of Fast R-CNN (13.6%).

Firstly, the YOLO algorithm has obvious advantages in speed. In fact, in addition to the real-time detection effect of YOLO, Faster R-CNN, SSD and other algorithms can also meet the requirements of real-time detection. However, we not only need to detect and classify the target, but also needs to perform complex functions such as tracking and license plate recognition. Therefore, it is hoped that the speed of the target detection and recognition algorithm can be as fast as possible.

Secondly, although the YOLO algorithm is fast, it has a shortcoming. It is not suitable for detecting overlapping objects. Of course, in order to achieve high FPS, such a sacrifice has to be made. But this does not have much impact on detecting vehicles parked on the roadside, because according to the angle at which we install the camera, there is no large-area overlap between vehicles. So, choose the YOLO algorithm, and its lower accuracy will not have much impact on vehicle detection. In addition, YOLO is based on Grid for target detect, each grid detects B targets, which is fixed, so YOLO has no advantage in detecting small targets. Then, the angle at which we install the camera and the distance between the camera and the target are determined, and this shortcoming of YOLO will not affect vehicle detection. So, after eliminating all the shortcomings, only the advantages are left.

Since the first times publication of YOLO, it has carried out four iterations of the version. Each iteration of the YOLO version has improved the YOLO algorithm itself[34,35]. It not only optimizes the network structure, but also applies some new technologies to it. Of course, the central idea remains unchanged, that is, "You Only Look Once". In addition, Ultralytics has launched YOLOv5. Although it is the fifth version, it has been controversial, but it has to be said that it has some improvements compared to YOLOv4.

2.2 Trapezoidal convolution algorithm

The shooting angle of the roadside camera generally adopts a certain top-down angle for shooting, rather than shooting perpendicular to the road surface, as shown in Figure 2.1.

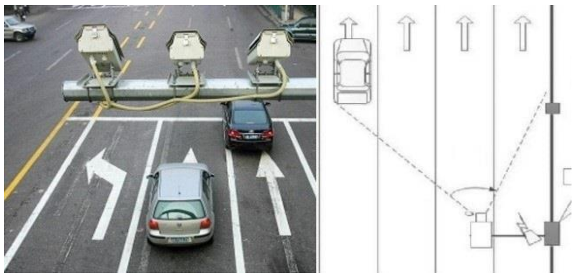


Fig 2.1 Shooting angle of roadside camera

The direction of the camera on the roadside is almost the same as the direction of the road, so there must be a smaller vehicle far away from the camera, and a larger vehicle closer to the camera. The characteristic of the image taken from this angle is that the vehicle at the top of the picture will be smaller, the vehicle at the bottom will be larger, and the rectangle in reality will be stretched into a trapezoid. As shown in Figure 2.2.

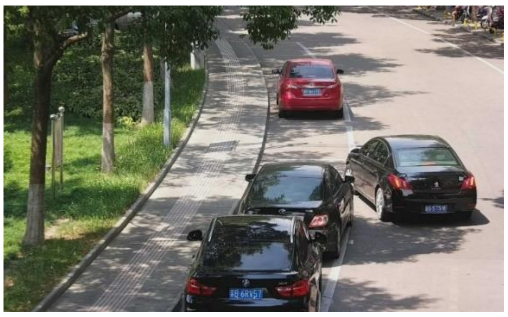


Fig 2.2 The rectangle in reality will be stretched into a trapezoid

It can be clearly observed that the more above the picture, the smaller the proportion of the vehicle in the picture. This shows a problem, in the same picture, the density of the objects will not be uniform. Since the density of the objects is not uniform, we will inevitably waste performance if we do the same processing on the positions of different densities.

According to this situation, this thesis proposes a trapezoidal convolution algorithm, which achieves the accuracy similar to YOLOv5 in the current scene, and at the same time the speed is improved.

2.2.1 The principle of trapezoidal convolution algorithm

The traditional convolution method is to perform the convolution operation by moving the convolution kernel horizontally or longitudinally by a fixed step. The convolution process is shown in Figure 2.3.

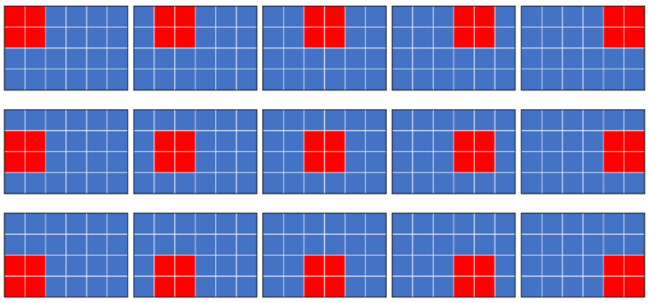


Fig 2.3 Traditional convolution method

The traditional convolution method uses a 2×2 filter with a step size of 1, and performs convolution in a picture with a height of 4 and a width of 6. 15 matrix multiplications are required.

The trapezoidal convolution algorithm achieves the purpose of reducing the number of convolutions by skipping some specified matrix multiplications. The trapezoidal convolution process is shown in Figure 2.4.

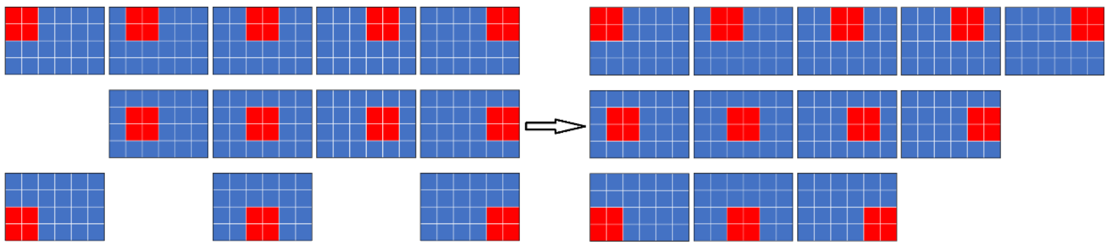


Fig 2.4 Trapezoidal convolution algorithm

In this algorithm, the number of convolutions required to perform a complete convolutional layer operation is 12 times, which reduces the amount of calculations by about 20%.

2.2.2 Positioning principle of trapezoidal convolution

Trapezoidal convolution can be realized because of the positioning principle of YOLO. The YOLO algorithm is based on CNN, and multiple convolutions in CNN will have a characteristic, which is called the invariance of spatial position information. Although the size of the picture is reduced during the convolution process, the position correspondence is preserved. As shown in Figure 2.5, the value of the area marked in red in the first picture will only affect the area marked in red in the second picture after the first convolution. The area marked in red on the second picture will only affect the area marked in red on the third picture. This is the positioning principle of YOLO.

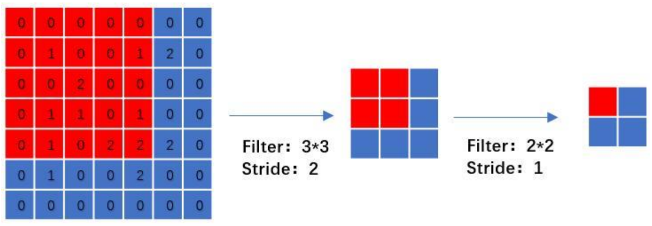


Fig 2.5 Positioning principle of YOLO: invariance of spatial position information

Trapezoidal convolution is based on this principle. Selectively skipping some convolution operations during the convolution process is equivalent to skipping some unimportant pixels in the picture. Because some unimportant convolution operations are skipped, in this way, the recognition accuracy can be guaranteed and the recognition speed can be improved.

2.2.3 Implementation of trapezoidal convolution algorithm

YOLOv5 uses *PyTorch* as a deep learning framework. The convolutional layer inside uses the *Conv2d* class, and the *Conv2d* class uses the *conv2d* function in *functional.py*. The implementation of this function depends on the *THNN* library. The implementation of matrix multiplication is specifically in the *SpatialConvolutionMM.c* file under the generic folder in the *THNN* library.

The forward propagation in *SpatialConvolutionMM.c* uses the *unfolded_copy* function, and its specific implementation is in the *unfold.c* file in the same level directory of *SpatialConvolutionMM.c*. To implement the trapezoidal convolution algorithm, the main thing is to modify the *unfolded_copy* function.

The principle of the *unfolded_copy* function is similar to the principle of *im2col*, and its main function is to convert convolution into matrix multiplication. Figure 2.6 is a simple example, where input and output $channel = batch = 1$, convolution kernel size $kH = kW = 3$, input size $inputHeight = inputWidth = 7$, stride size $dW = dH = 2$, padding size $padW = padH = 1$.The output: $outputHeight = outputWidth = 3$ can be calculated.

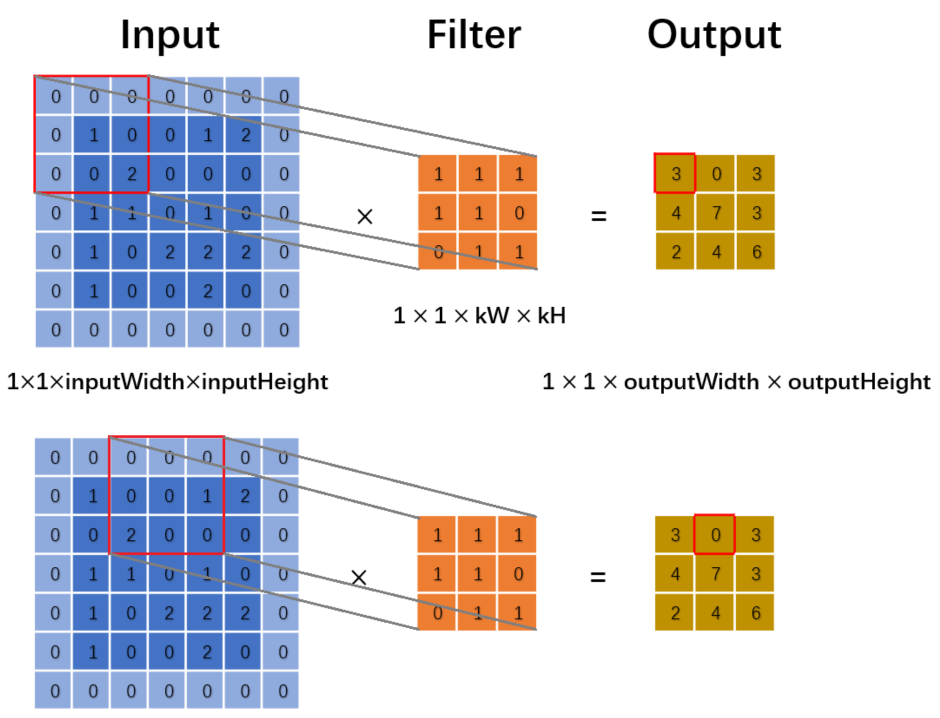

Input
Filter
Output
1 × 1 × kW × kH
1×1×inputWidth×inputHeight
1 × 1 × outputWidth × outputHeight

Fig 2.6 Convolution example

To turn convolution into a matrix multiplication operation, need turn each convolution window in the input into a separate column, which is what *unfolded_copy* does, as shown in Figure 2.7.

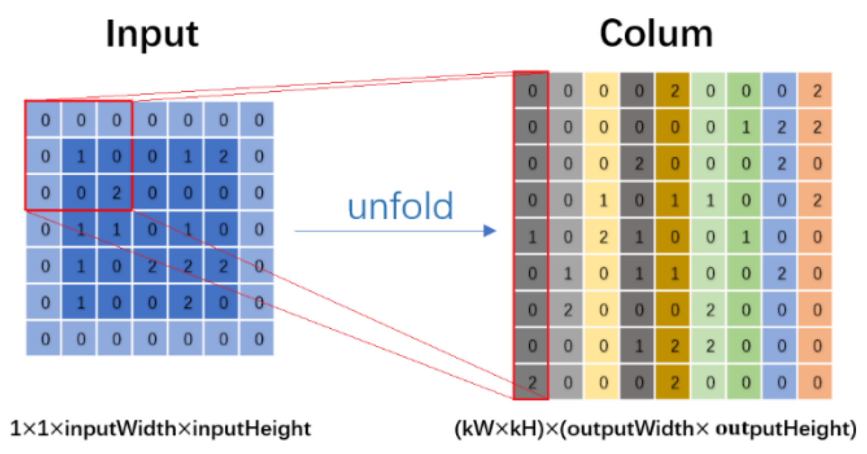

Input
Colum
unfold
1×1×inputWidth×inputHeight
(kW×kH)×(outputWidth× outputHeight)

Fig 2.7 Turn convolution into a matrix multiplication operation

Each column of the matrix on the right in Figure 2.7 is a convolution window of the original input matrix. The size of the converted matrix is $(nInputPlane \times kH \times kW) \times (outputHeight \times outputWidth)$. Resize into $(nOutputPlane) \times (nInputPlane \times kH \times kW)$ and you can directly multiply the matrix by the kernel to get the convolution result, as shown in Figure 2.8.

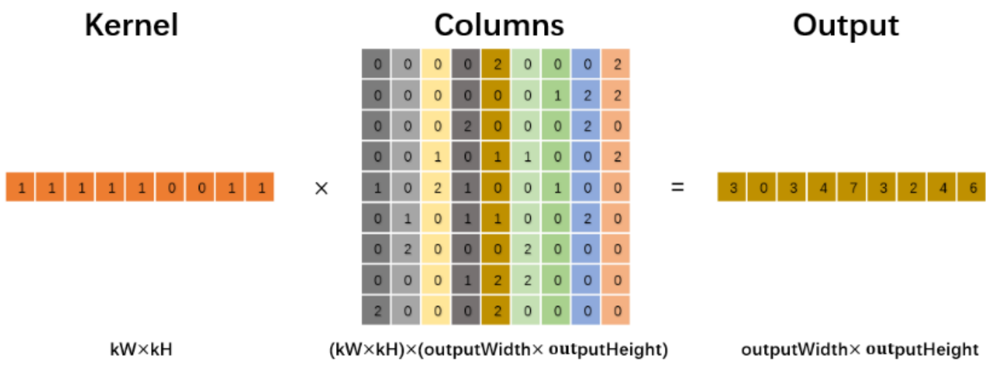

Kernel
Columns
Output
kW×kH
(kW×kH)×(outputWidth× outputHeight)
outputWidth× outputHeight

Fig 2.8 Directly multiply the matrix by the kernel

To implement the trapezoidal algorithm based on the principle of the *unfolded_copy* function, we only need to trim the columns after unfold, remove the unnecessary columns without affecting the accuracy of the entire detection algorithm, and then perform matrix multiplication with the kernel. For example, if the third and seventh columns of columns are removed here, the convolution result is shown in Figure 2.9:

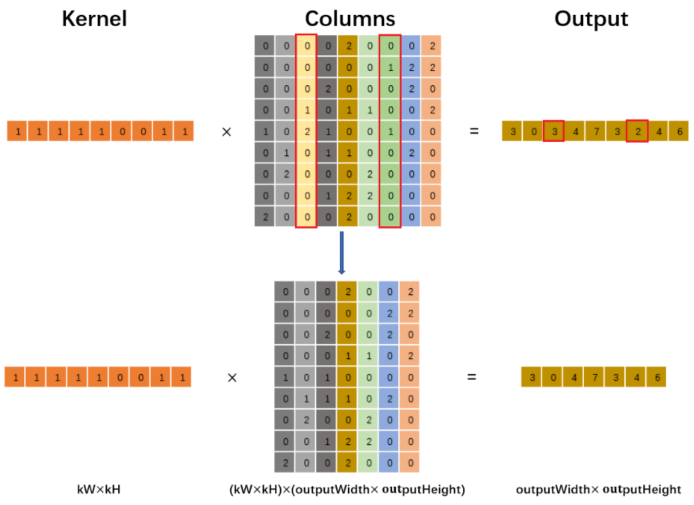

Kernel
Columns
Output
kW×kH
(kW×kH)×(outputWidth× outputHeight)
outputWidth× outputHeight

Fig 2.9 Example of removing columns

Originally, 9 matrix multiplication operations are required, but only 7 operations are required after cropping, which reduces the amount of calculations. But at the same time, we need to pay attention to the change in the dimension of the output. At this time, there are 7 columns after the output, so that you cannot restore the 3×3 matrix, which will affect the subsequent operations. In order not to affect the subsequent calculations, here is to fill in the places that were originally lost, and the filled value uses the average value of its adjacent positions, as shown in Figure 2.10.

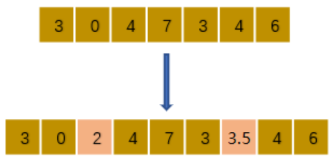


Fig 2.10 Filled value uses the average value of its adjacent positions

2.2.4 Application in actual scene

The input image size used in this paper is *640*. In the convolution of the first row, the convolution is skipped once, the convolution in the second row is skipped twice, and skip *N* convolutions in the convolutions of the last row. So, when detecting a picture, the total number of skipped convolutions is *1+2+...+N* times. The following is the speed comparison when trapezoidal convolution is not used and when trapezoidal convolution is used. Among them, a 3-channel RGB image with a size of 640×640 pixels are used, which is divided into 12 channels 320×320 by the *Focus* algorithm in *YOLOv5*. The size of the convolution kernel is 3×3, and the number of convolution kernels is 32. At the same time, the convolution process is looped 100 times. The reason for this setting is to be close to the algorithm complexity of the *yolov5s* model's recognition process. In addition, in order to make the results more comparable, the same weight is used when creating *Conv2d* objects in PyTorch.

Table 2.1 Comparison of trapezoidal convolution and traditional convolution

Devices	Methods	Speeds (ms)
CPU	Traditional Convolution	983
	Trapezoidal Convolution	843
GPU	Traditional Convolution	8
	Trapezoidal Convolution	8

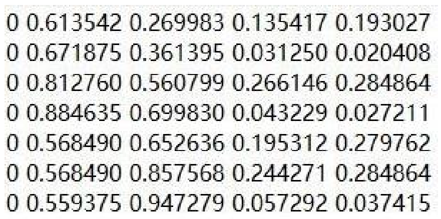
As can be seen from Table 2.1, although the performance of trapezoidal convolution on GPU is not better than traditional convolution, on CPU, trapezoidal convolution has a significant improvement in speed compared to traditional convolution.

2.3 Model training

2.3.1 Dataset introduction

This dataset are video collections of the parking behavior of vehicles on multiple road sections, and randomly intercepts 2200 pictures from each road section videos, 2000 pictures as the training set, and 200 as the validation set. Then use the annotation tool to annotate the images.

The dataset format of each image is shown in Figure 2.11.



```
0 0.613542 0.269983 0.135417 0.193027
0 0.671875 0.361395 0.031250 0.020408
0 0.812760 0.560799 0.266146 0.284864
0 0.884635 0.699830 0.043229 0.027211
0 0.568490 0.652636 0.195312 0.279762
0 0.568490 0.857568 0.244271 0.284864
0 0.559375 0.947279 0.057292 0.037415
```

Fig 2.11 Dataset format

Each frame of picture corresponds to a txt file, and one line in each file represents a car. There are several cars in a picture, and there are several lines in the corresponding txt file. In each row, the first number represents the category. In this chapter, the category is only car, so the number is always 0. The following four numbers respectively represent the ratio of the abscissa of the center of the Bounding box to the width of the image x, the ratio of the center ordinate to the image height y, the ratio of the bounding box width to the image width w, and the ratio of the height of the bounding box to the image height h.

2.3.2 Model training

When we use the yolov5s model, we found that it takes about 0.1 seconds to infer an image under the CPU, which obviously cannot meet the needs of real-time detection, so we simplified the yolov5s model. Then we trained this model for 200 epochs with our dataset. The loss during training is shown in Figure 2.12.

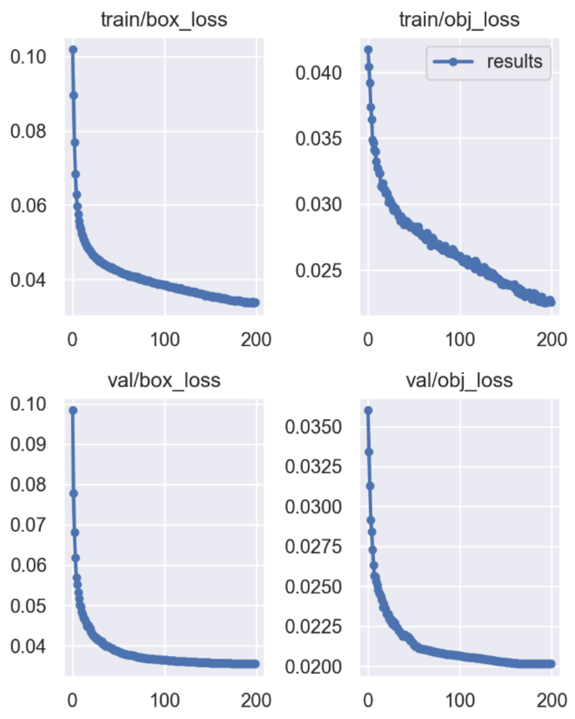


Fig 2.12 Train and valid loss

It can be seen that when the training iteration reaches about 200, the validation loss of the model tends to be stable.

As can be seen in Figure 2.13, the precision and recall values of the model continue to increase during the training process, and the model's effect is getting better and better.

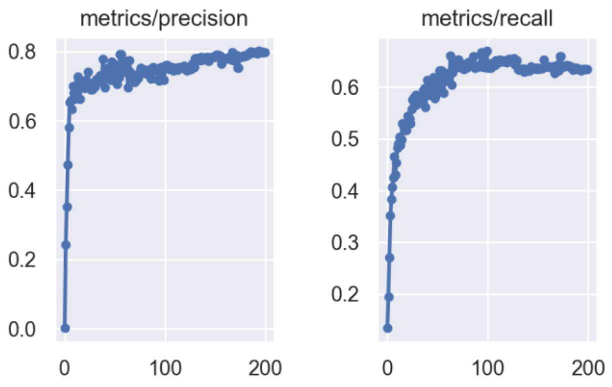


Fig 2.13 precision and recall curves

Figure 2.14 shows the PR curve when the IOU threshold is set to 0.5 during model training.

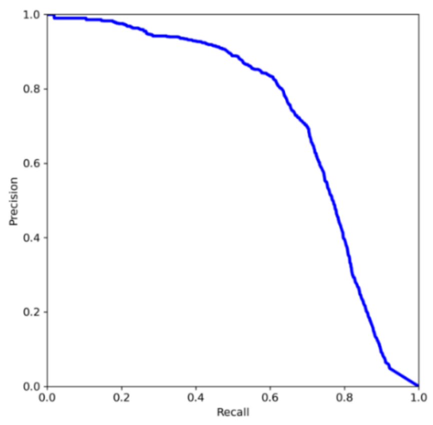


Fig 2.14 PR curve

The AP value is shown in Figure 2.15. Because there is only one category in this model, mAP is equal to the AP value. After training, mAP@0.5 is 0.722 and mAP@0.5:0.95 is 0.485.

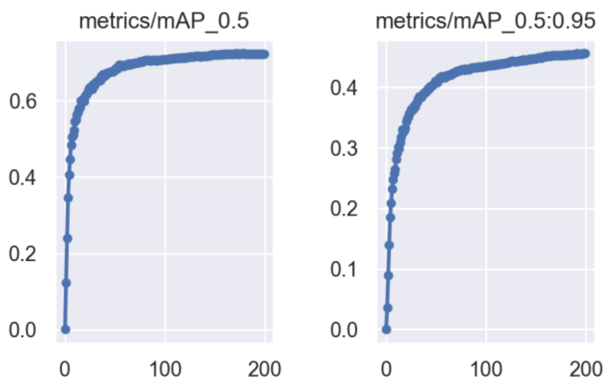


Fig 2.15 mAP@0.5 and mAP@0.5:0.95

2.3.3 Experimental result

In order to verify the effectiveness of trapezoidal convolution, we trained the simplified yolov5s model using traditional convolution in the same way, and then compares the two models. The GPU uses NVIDIA RTX GeForce 2060 and the CPU uses Intel core i5-11400, the number of vehicles contained in the test image is 6. The model comparison results are shown in Table 2.2:

Table 2.2 Comparison of trapezoidal model and traditional model

Methods	GPU Speed (s)	CPU Speed (s)	mAP@0.5	mAP0.5:0.95
Trapezoidal	0.018	0.046	0.727	0.486
Traditional	0.018	0.054	0.731	0.489

It can be seen that the use of trapezoidal convolution can increase the inference speed of the model on the CPU while losing a small amount of accuracy. It can be seen from the table that the accuracy loss is about 0.55%, and the inference time under the CPU is reduced by about 14%, so the speed improvement effect of the trapezoidal convolution on the CPU is obvious.

2.4 Summary

This chapter mainly introduces the link between vehicle detection and recognition in this thesis in detail. First, it introduces the current situation of object detection and recognition, and then introduces the application scenarios of this thesis and the choice and reason of the basic algorithm. At the same time, this paper proposes the trapezoidal convolution algorithm, which aims to improve the inference speed of the model and introduces its working principle in detail. Finally, this chapter uses its dataset to train and analyze the effect of the model that uses trapezoidal convolution and then compares it with the model that does not use trapezoidal convolution. While the speed of the trapezoidal convolution model is improved, the accuracy of inference is reduced not much.

3 License plate recognition and binding

There is a one-to-one correspondence between vehicles and license plates. Each vehicle needs to bind its own license plate to identify the vehicle being tracked. When a vehicle is marked with a license plate, as long as the vehicle is still being tracked, no matter whether the license plate of the vehicle is blocked or not, the program can still remember the license plate of the vehicle and record the parking and leaving behavior of the vehicle.

After the vehicle recognition process, we have got the Bounding Box of the vehicle. In the process of vehicle recognition, we compressed the original image to improve the efficiency of vehicle recognition. However, in the recognition of the license plate number, if the compressed image continues to be used, the accuracy of the license plate number recognition will be reduced. Therefore, it is necessary to map the compressed image to the original image, map the Bounding Box of the car to the original image, and crop it out to recognize the license plate number.

3.1 Bounding box location mapping

In YOLOv5, the image resize process is like this:

(1) Calculate the zoom ratio:

$$r = \min\left(\frac{i}{h}, \frac{i}{w}\right) \tag{3.1}$$

Divide the set size *i* by the height *h* and the width *w*, and take the smaller value as the scaling factor *r*.

(2) Calculate the scaled size:

$$H = h \times r \tag{3.2}$$

$$W = w \times r \tag{3.3}$$

Multiply the height *h* and width *w* of the original image by the zoom factor to obtain the zoomed length *H* and width *W*.

(3) Calculate the black border filling value:

When $\frac{i}{h} > \frac{i}{w}$,

$$B = \frac{np.\mathrm{mod}\left(w - W, 64\right)}{2} \tag{3.4}$$

When $\frac{i}{h} < \frac{i}{w}$,

$$B = \frac{np.\mathrm{mod}\left(w - W, 64\right)}{2} \tag{3.5}$$

When the zoom factor of *h* is greater than the zoom factor of *w* (ie *h*<*w*), black borders are added to the left and right sides of the image, and the width of the black border is *B*. When the zoom factor of *w* is greater than *h* (ie *h*>*w*), black borders are added to the upper and lower sides of the image, and the height of the black border is *B*.

In YOLOv5, in order to make the height and width of the input image a multiple of 32, black borders are added on the top and bottom or left and right sides of the picture. Therefore, if we directly map the car's Bounding Box according to the ratio of the compressed image to the original image, the cropped image will have some distortion. Although the slight distortion of the image has little effect on the experimental results, it can be avoided why not avoided. So here, by performing the reverse operation of the process in YOLOv5, taking the Bounding Box of the license plate after target recognition as input, we can get the corresponding cutout of the bounding box of the car in the original image. The specific process is as follows:

(1) Calculate the zoom factor to determine the direction of black border addition:

$$r = \min\left(\frac{i}{h}, \frac{i}{w}\right) \tag{3.6}$$

(2) Remove the black border;

(3) Scaling Bounding Box:

$$X_b = \frac{x_b}{r} \tag{3.7}$$

$$Y_b = \frac{y_b}{r} \tag{3.8}$$

$$W_b = \frac{w_b}{r} \tag{3.9}$$

$$H_b = \frac{h_b}{r} \tag{3.10}$$

According to the zoom factor, find the corresponding position and size of the Bounding Box in the original image.

3.2 License plate recognition

There are many researches on the recognition of license plate numbers. LIU Jing-yu, LIU De-er[36] divides license plate recognition into four parts: license plate position, tilted license plate correction, character segmentation, and character recognition.

LIU Jing-yu locates the license plate based on the blue background color of the license plate. Although this method is simple, the effect is not good, especially when the light contrast is not strong. BI Bo, SHAO Yong-qian[37] also used the color of the license plate to locate the license plate, but he also added the font color, license plate aspect ratio and other license plate features to improve the accuracy of the license plate location. WU Anhui, HE Qili[38] uses the U-Net model to locate the license plate.

For tilting license plate correction, Liu Jing-yu and BI Bo both use the Hough line detection algorithm to calculate the tilt angle of the license plate, and then correct the license plate through affine transformation. This is the more commonly used method at present. It's just that when we use the Hough line detection algorithm, we need to set some hyper parameters. The same set of hyper parameters has different effects in different scenarios. WU Anhui uses U-Net to identify the position of the license plate and then rotates the license plate.

For character segmentation, LIU Jing-yu proposed that the size of the license plate is $440mm \times 140mm$, and she divides the characters by calculating the proportion of each character. BI Bo judges the character position by counting the pixels in the license plate. WU Anhui's paper did not mention how to perform character segmentation.

About license plate recognition, the previously used directions include feature matching, SVM, KNN, etc. Now, more methods are used to recognize characters based on CNN.

3.2.1 One-step license plate recognition algorithm

The scheme adopted in this thesis for the whole license plate recognition is different from the scheme mentioned above. Some of the schemes in the front are either not effective or too complicated to implement. This thesis intends to integrate license plate correction, character segmentation, and character recognition into one step.

3.2.2 YOLOv5-based license plate recognition

Here, YOLOv5 is still used as the basic object detection network to recognize the characters of each license plate. In this way, steps such as license plate correction and character segmentation are eliminated.

The specific implementation process is as follows:

First, the vehicle images in the dataset are cropped to imitate a single vehicle that has been cropped after vehicle detection. After collecting such dataset, use the labeling tool to frame each license plate character, as shown in Figure 3.1.

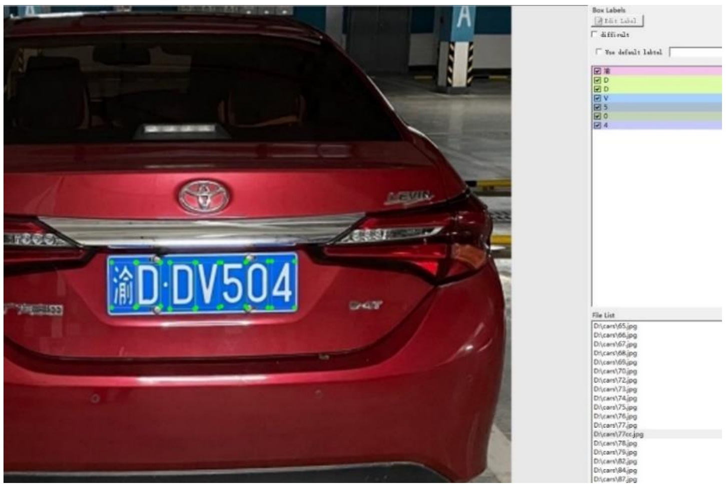


Fig 3.1 Labeling tool

The labels are capital letters A-Z (not including I, O), numbers 0-9, and the abbreviation of each province. A total of 65 labels, that is, a total of 65 classes.

3.2.3 Model training and result

The dataset used in this model training is cropped from the reverse mapping of the pictures after resizing in Chapter 2, and then these pictures are manually annotated. The dataset has a total of 1001 images, including 901 in the training set and 100 in the validation set. The network model uses the simplified yolov5s model with traditional convolution mentioned at Chapter 2, and the PR curve of the model training is shown in the figure 3.2:

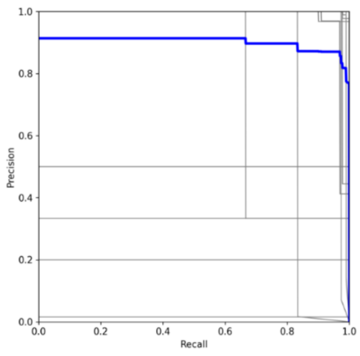


Fig 3.2 PR curve of training

The result of this model is shown in Table 3.1.

Table 3.1 result of HVS model

GPU Speed (ms)	CPU Speed (ms)	mAP@0.5	mAP@0.5:0.95
18	54	0.898	0.697

After training, the effect of the model is shown in Figure 3.3.

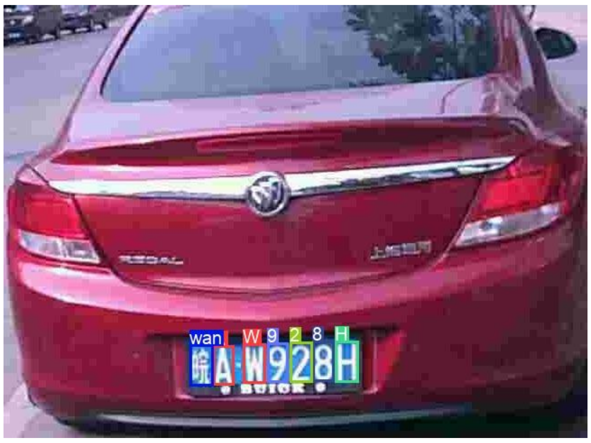


Fig 3.3 result after detection

3.2.4 License plate information verification

The license plate number recognition may be inaccurate due to some special scenes, such as bright or dark light, unclear license plate information on rainy days, and insufficient image resolution caused by the vehicle being too far away from the camera. At this time, some simple algorithms can be added to eliminate some recognition errors.

When it is found that there is an error in the recognition of the license plate number information, the matching of the license plate and the vehicle can no longer be performed, which improves accuracy.

3.2.4.1 Fixed number of characters

First, in a bounding box containing a car, there must be only one license plate. A license plate has only 7 characters. So, if the number of recognized characters is not 7, then the result of this recognition is a failure. Therefore, the function of counting the number of characters is added to the program to determine the accuracy of this recognition.

3.2.4.2 The centers of the characters are on the same line

If the license plate is correctly identified, the center of each character contained in it is almost on the same straight line, so this article uses the distance from the point to the straight line to judge whether the characters are almost on the same straight line. If it is large, it is considered that the recognition has failed this time. The verification formula is as follows:

First, find the unary linear regression equation of these characters: $\mathrm{Y}_t = kx_t + b$, and

$$\begin{cases} k = \dfrac{7\sum x_i Y_i - \sum x_i \sum Y_i}{7\sum x_i^2 - \left(\sum x_i\right)^2} \\ b = \dfrac{\sum Y_i}{7} - a\dfrac{\sum x_i}{7} \end{cases} \tag{3.11}$$

Then, according to the unary linear regression equation, find the square of the distance from the center of each character to the unary regression line, and sum:

$$S = \sum \frac{\left(kx_j - Y_j + b\right)^2}{k^2 + 1} \tag{3.12}$$

By setting the threshold of S, it is determined whether the centers of the characters are approximately in the same straight line.

3.2.4.3 The distance between characters is approximately equal

If the recognized license plate information is correct, then the distance between two adjacent license plates in the license plate is approximately equal. In order to detect whether the distance between two adjacent license plates is equal, the following algorithm is used for verification:

- Sort the characters in the license plate. Each character in the license plate contains the starting abscissa *x*, the starting ordinate *y*, the width *w* of the character's bounding box, and the height *h* of the bounding box. Here, the starting abscissa *x* is used for sorting;
- Calculate the distance between adjacent characters and calculate the variance of the distance;
- It is judged whether the variance exceeds the set threshold, if it exceeds the preset threshold, it is considered that the license plate detection is wrong this time.

3.3 License plate number binding

When a vehicle is being tracked, it can be judged whether there is license plate information in the currently tracked vehicle. If it exists, the matching algorithm is used to calculate whether the license plate number information belongs to the vehicle. If the license plate belongs to the vehicle, the license plate number information is recorded in the tracker corresponding to the vehicle. If it does not exist, the license plate information will not be recorded and continue to wait until the license plate number information is obtained. All these associated operations are based on the Tracker class designed in this paper. The relationship between the Tracker class and the detected vehicle (detection) and license plate is shown in Figure 3.4.

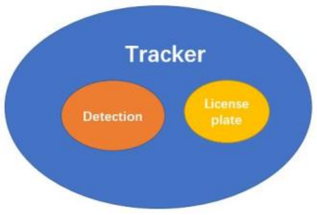


Fig 3.4 The relationship between the Tracker, detection and license plate

3.3.1 Binding and update of license plates

The binding process of the vehicle and the license plate is shown in list 3.1.

List 3.1 Binding and update of license plates

```
1: Bind Tracker and Detection
2: Cut out the vehicle according to the reverse mapping formula
3: Realize license plate recognition through YOLOv5
4: License plate verification
5: If there are multiple license plates, select the license plate based on the license plate confidence
6: Update or save the license plate, save the license plate information in the Tracker, if the license plate information already exists, compare the confidence of the new and old license plates, and keep the license plate with higher confidence.
```

3.3.2 Overlapping vehicle license plate information attribution

Vehicles on the road often overlap, and their bounding boxes will also overlap, especially when the prediction of the bounding box is not so accurate. The overlapping state is shown in Figure 3.5.

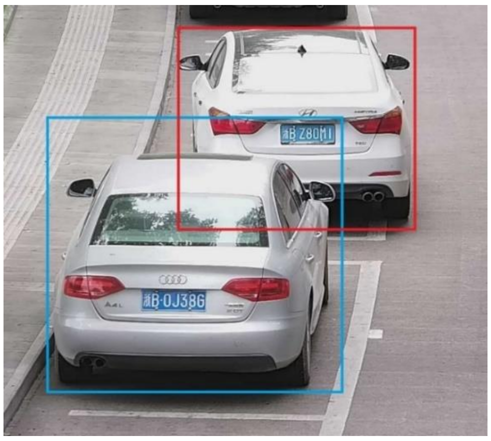


Fig 3.5 Vehicle overlap

At this time, the bounding boxes of the two vehicles occupy one license plate at the same time. At this time, we need to consider which vehicle the license plate belongs to. It is easy for human eyes to know that the license plate belongs to the car that is not blocked. The key is how to let the program know which car is less blocked. Fortunately, through experimental statistics, it is found that for a vehicle surrounded by a blue frame, the average confidence of its own license plate is higher than the confidence of other vehicle license plates. The reason for this phenomenon is that there is a correlation between the position of the license plate and the vehicle, so the confidence of the license plate of the non-own car will be lower than the original license plate of the car. So, when a bounding box analysis gets two license plates, the confidence of the license plate is used to determine which license plate number information belongs to that vehicle, and then the license plate information is recorded in the Tracker. In addition, through the license plate verification in the previous section, some invalid license plates can also be eliminated for the current vehicle.

3.4 Summary

This chapter first introduces the current research status of license plate recognition. The current research on license plate recognition still focuses on positioning first and then recognition. Then according to the current status of license plate recognition, this thesis proposes a one-step license plate recognition scheme through YOLOv5 and achieved good results. Finally, it introduces how to realize the association between the license plate and the vehicle through the program, and solves some problems encountered in the license plate association.

4 Vehicle tracking

In the previous chapter, we extracted the position information of the vehicle in each frame of image, and its output will be used as the input to the tracking algorithm. After we extract the target, we also need to link each frame of image, which car A in this frame corresponds to which car in the previous frame. The purpose of this is that as long as we successfully identify the license plate number of car A in a certain frame, then in the following video, when the license plate number of car A is blocked, we still know that it is car A and know its license plate number.

4.1 Overview of tracking algorithms

There are many tracking algorithms[39-46], such as Li Muzi[47] in his thesis, also uses the output of the YOLOv5 algorithm as the input of the tracking algorithm, and then he uses the KCF[48] algorithm to track the objects, and its detection objects are pedestrian, so the KCF algorithm Has a good effect. But this article is to detect the parking behavior of vehicles. One of the problems that parking behavior must face is that the vehicle will turn, and the turning will cause deformation, and KCF can't deal with the scene where the target scale and shape change well.

In addition to the KCF algorithm, there are SORT, Deep SORT, MOTDT[49], etc.

4.2 Deep SORT

After the YOLOv5 algorithm was proposed, the SORT algorithm was proposed. Its function is to track the detections output by YOLOv5. Just like the name of SORT, it is a simple algorithm, mainly using the Kalman filter and the Hungarian algorithm. The Kalman filter algorithm is divided into two steps: the first step is to predict, through the position and speed of the object in the previous frame, the position and speed of the object in the next frame are predicted; the second step is to update, the predicted value of the normal distribution and the observation value of the normal distribution are linearly weighted to obtain the current state of the system prediction. The role of the Hungarian algorithm is to match the detections in the two frames, and according to the matching results, three states of Unmatched Tracks, Unmatched Detections, and Matched Tracks are obtained.

After SORT, Nicolai Wojke, Alex Bewley, and Dietrich Paulus proposed the Deep SORT algorithm. Deep SORT draws on the main framework of SORT, but it is much more complicated than SORT. Its biggest feature is the addition of appearance information. It uses models from the ReID field to extract features, reducing the number of ID switches. The flowchart of Deep SORT is shown in Figure 4.1.

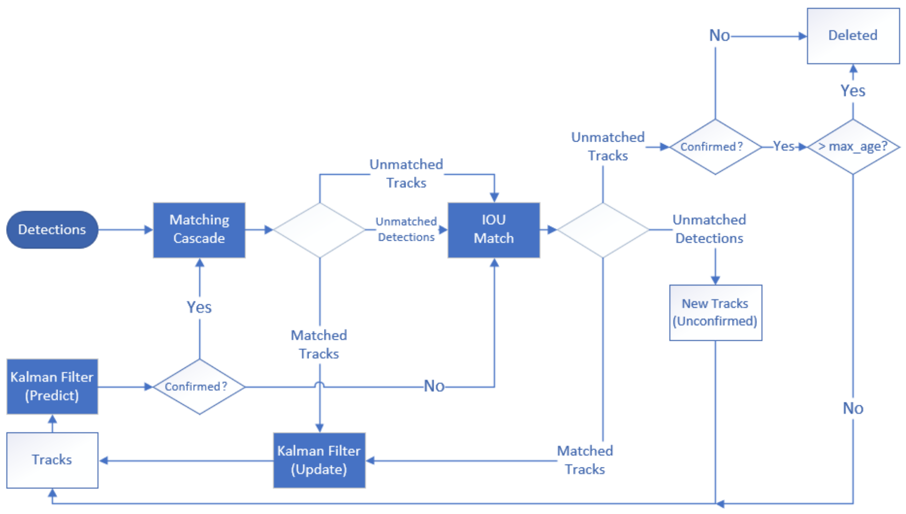


Fig 4.1 Flowchart of Deep SORT

4.2.1 Trajectory processing of deep SORT

Trajectory processing mainly refers to when the trajectory is terminated and when a new trajectory is generated. First, there is a threshold named *max age* for each track to record the time period from the last successful match to the current match. When the waiting time for successful matching is greater than *max age*, it is considered that the object associated with this trajectory is no longer in the field of view, and then mark it as *deleted*. If the successful matching time is less than *max age*, continue tracking. For detections that do not match successfully, mark the trajectory as *tentative* and continue to observe several frames. If the consecutive match is successful, mark it as *confirmed*, otherwise mark it as *deleted*.

4.2.2 Kalman filter

In deep SORT, the role of Kalman Filter is trajectory prediction. The estimated tracks at the current moment are obtained from the tracks at the previous moment after passing through a Kalman Filter, and then matched with the Detections at the current moment. If the match is successful, the tracks and their corresponding Detections are bound respectively, and at the same time, the current tracks are updated according to the properties of Detections and tracks.

Kalman Filter's state estimation: Use an 8-dimensional tensor to describe the state of the trajectory features at a certain moment. The speed information v_x, v_y, v_a, v_h corresponding to the 4 variables in the image coordinates. Among them, the first four variables are observation variables, and these four variables are used as input, and then the estimated abscissa position x', center ordinate position y', aspect ratio a', height h', and v_x', v_y', v_a', v_h' are obtained. At the same time, an 8×8 covariance matrix will be returned.

4.2.3 Reid model

There are two main differences between Deep SORT and SORT. One is the addition of matching cascade, and the other is the addition of life management cycle. The extra part is the part marked by the red box in Figure 4.2.

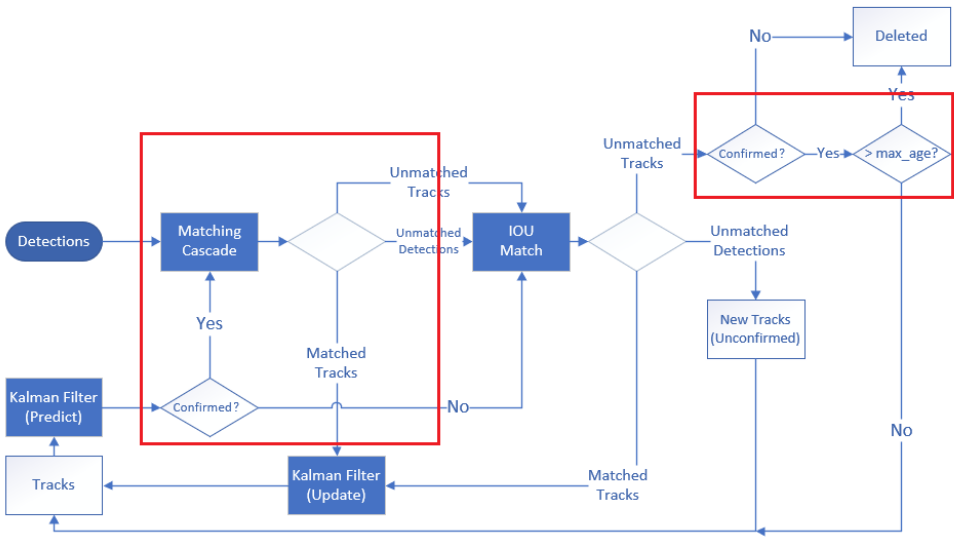


Fig 4.2 The difference between Deep SORT and SORT

In SORT, after Kalman Filter filtering, IOU Match is performed directly, and there is no cascade matching step. After IOU Match, there is no life cycle management after the track is not matched, but the track is deleted directly. These two steps greatly reduce the probability of ID switch and solve the problem of occlusion. Here we will talk about the Reid network model in cascade matching, because this is the focus of this chapter to optimize. The Reid network model in the deep SORT thesis is shown in Table 4.1.

Table 4.1 Reid model in Deep SORT

Name	Patch Size/Stride	Output Size
Conv1	$3 \times 3/1$	$32 \times 128 \times 64$
Conv2	$3 \times 3/1$	$32 \times 128 \times 64$
Max Pool 3	$3 \times 3/2$	$32 \times 64 \times 32$
Residual 4	$3 \times 3/1$	$32 \times 64 \times 32$
Residual 5	$3 \times 3/1$	$32 \times 64 \times 32$
Residual 6	$3 \times 3/2$	$64 \times 32 \times 16$
Residual 7	$3 \times 3/1$	$64 \times 32 \times 16$
Residual 8	$3 \times 3/2$	$128 \times 16 \times 8$
Residual 9	$3 \times 3/1$	$128 \times 16 \times 8$
Dense 10		128
Batch and ℓ_2 normalization		128

The network model finally outputs a 128-dimensional feature vector. It has a total of 2,800,864 parameters. When using NVIDIA GeForce GTX 1050 mobile GPU and 32 bounding boxes, it takes about 30ms to perform a forward propagation operation.

4.3 Model optimization

4.3.1 Input of model

From the vehicle detection and recognition, we have obtained the position information of the vehicle in the image, as well as the position information of the license plate. But here, we don't need the location information of the license plate, what we need is the location information of the vehicle. We use the vehicle location information to crop these vehicles from the image, and send the cropped parts to the ReId network model for feature extraction, and finally a 128-dimensional feature vector is obtained.

4.3.2 Realistic scene

The system proposed in this thesis is adapted to the actual scene where a camera monitors the road and collects video. In this scenario, we have two points to note:

The first point is that the only object detected in our object detection and recognition is vehicle. This shows that our optimized ReId network model should be able to extract good feature vectors for vehicles, rather than for other categories like person. The ReId model used by Deep SORT is for the Person, so we need to modify the model to make it more suitable for vehicles.

The second point is that the field of view of a camera placed on the roadside is limited, and the number of vehicles it can capture in one frame of video will not exceed twenty. This means that the final feature extracted by our network model is sufficient as long as it can be used to correctly distinguish 20 vehicles. When the conditions are met, the fewer parameters of the model, the better, because the fewer parameters indirectly mean the faster the speed can be.

4.3.3 Horizontal and vertical separation network model

At present, the research on pedestrian re-identification has been very much and very in-depth, and the research on vehicle re-identification is not as rich as the research on pedestrian re-identification. But there still are some papers[50-53].

The difference between vehicle re-identification and pedestrian re- identification is that the human body is vertically symmetrical and can be divided into head, trunk, legs and feet along the height dimension, making the height division more practical. The front and rear views of the car are also vertically symmetrical. But the side view is asymmetrical and can be roughly divided into engine hood, door, trunk, etc. Along the horizontal axis, in a multi-view vehicle re-identification dataset, both height and width partitions are beneficial. When the vehicle is parked on the side, the side of the vehicle generally faces the camera, which is more conducive to collecting the feature of the side of the vehicle.

In a general CNN network, the feature extraction process is completed by convolution. When the convolution kernel is small, some redundant features may be extracted, as shown in Figure 4.3.

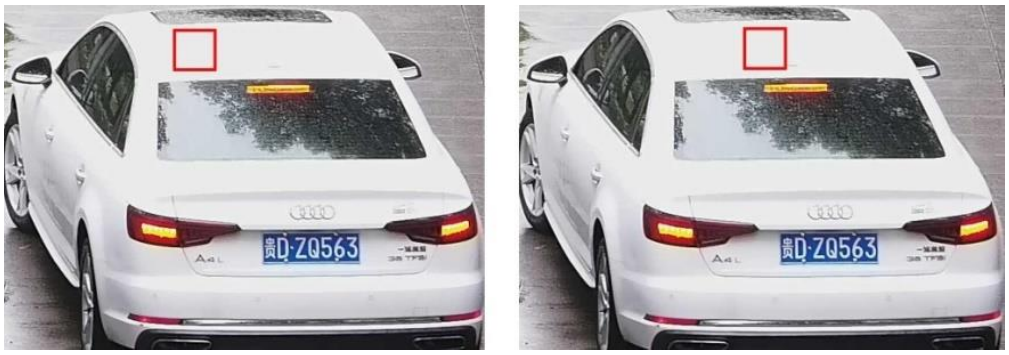


Fig 4.3 Redundant features in CNN network

From Figure 4.3, we can see that ordinary convolution will extract the features of each position one by one, but for vehicles, there are not many textures inside the vehicle. The textures are mostly concentrated on the contour, and all the horizontal and vertical convolutions are performed. There will be redundant features.

Based on the above analysis, this thesis proposes a feature extraction network focusing on horizontal and vertical. The structure of Horizontal and vertical Separation (HVS) network is shown in the table 4.2. It separates and extracts image features horizontally and vertically. Compared with normal convolution, it is shown in Figure 4.4. It aims to reduce the parameters of the network model and increase the speed of the network to extract features when the network extracts effective features.

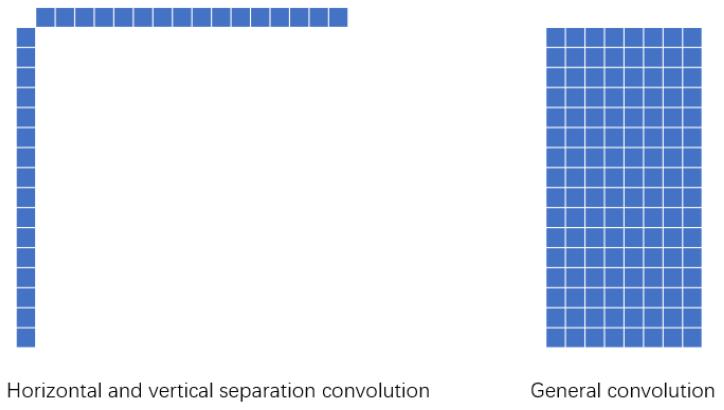


Fig 4.4 Comparison between HVS and General

The parameters of HVS network are shown in Table 4.2.

Table 4.2 Horizontal and vertical separation network

Name	Patch Size/Stride	Output Size
Conv1	$3 \times 3/1$	$32 \times 128 \times 64$
Conv2	$3 \times 3/1$	$32 \times 128 \times 64$
Conv3_1	$128 \times 3/1$	$32 \times 1 \times 64$
Conv3_2/Transpose	$3 \times 64/2$	$32 \times 1 \times 64$
Cat (Conv3_1, Conv3_2/Transpose)		$64 \times 1 \times 64$
Residual 4	$3 \times 3/1$	$64 \times 1 \times 64$
Residual 5	$3 \times 3/1$	$64 \times 1 \times 64$
Residual 6	$3 \times 3/2$	$64 \times 1 \times 32$
Residual 7	$3 \times 3/1$	$64 \times 1 \times 32$
Residual 8	$3 \times 3/2$	$128 \times 1 \times 16$
Residual 9	$3 \times 3/1$	$128 \times 1 \times 16$
Dense 10		128
Batch and ℓ_2 normalization		128

4.3.4 Model training

The model training uses the VeRi[54] dataset, which takes 776 cars at different times, different road sections, different angles, and different lighting, and has a total of 49,357 images. In this experiment, the 49357 images were first classified by the car number. Move them to 776 folders, each of which represents a certain car. Then randomly select

20 folders from these 776 folders for model training. Finally, randomly divide each folder into a training set and a validation set at a ratio of 19:1.

In the process of training the model, it is difficult to directly determine the quality of the feature extraction model, so this experiment adds a fully connected layer to the last layer of the feature extraction network model for classification. We can judge the quality of the feature extraction model by the quality of the classification result, because the function of the feature extraction model in this thesis is to distinguish the difference between each vehicle in the camera. If the model can extract some unique features of each vehicle well, it can distinguish each vehicle well.

The input image of the network model will be resized to 128×64, and the number of channels is 3. Batch size is set to 8; epoch is set to 60; the initial learning rate is set to 0.01, and the learning rate is reduced by 10 times after every 10 epochs; SGD optimizer is used; the dropout probability set in the dropout layer of Dense 10 Is 0.6; the loss function uses Cross Entropy Loss function:

$$H(p,q) = -\sum_{x}\left(p(x)\log q(x)\right) \tag{4.1}$$

The effect of the model training process is shown in Figure 4.5.

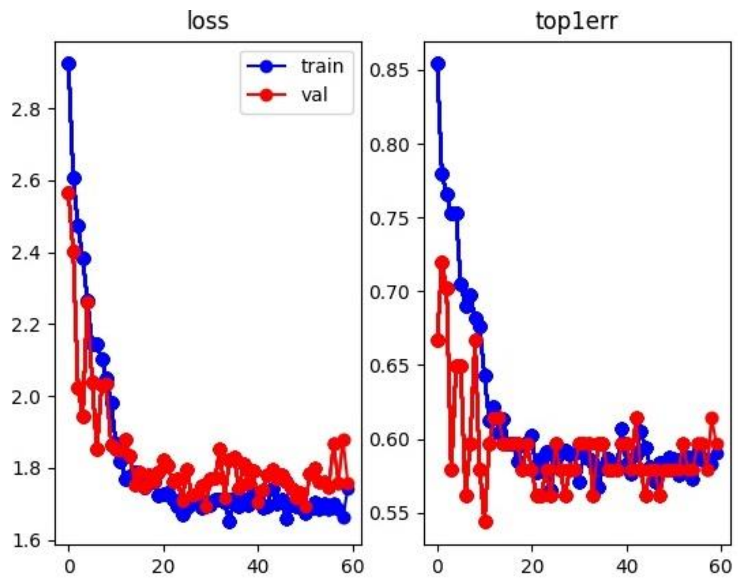


Fig 4.5 Model training process

4.3.5 Model comparison

Table 4.3 Comparison between Original model and HVS model

Model	Memory Size (KB)	Speed (ms)	Top1 Accuracy (%)
Original model	11287	36	59.8
HVS model	6659	25	56.3

The comparison result is compared with the optimal results obtained after multiple trainings by randomly extracting 20 categories from the VeRi dataset. Both the Original model and the HVS model have been trained no less than ten times, and then the best Top1 Accuracy is recorded in the table 4.3.

From the comparison results, it can be seen that although the accuracy of the HVS model proposed in this thesis is nearly 3 percentage points lower than that of the original model, its speed has increased by about 30%. And the memory size occupied by the model is also reduced by about 40%. In addition, putting the HVS model into deep SORT to track the vehicle, the effect achieved is almost the same as the original model.

4.4 Summary

This chapter mainly explains vehicle tracking in vehicle parking behavior recognition. First, it roughly describes the current research status in the field of object tracking and then focuses on the related content of deep SORT, a representative tracking framework with good performance. After in-depth understanding and research on the deep SORT framework and its Reid network model, this paper proposes a network model HVS model suitable for the vehicle parking behavior recognition scene in this thesis. After training the model, we got a model that slightly reduces the accuracy, but greatly improves the speed and reduces the memory size.

5 Vehicle trajectory classification

After obtaining the trajectory of the tracked vehicle, the behavior of the tracked vehicle can be classified. According to the classification result, it can be known whether the current behavior of the vehicle is parking, leaving, or no behavior. There are many ways to classify trajectories, but considering the real-time, multi-targets, uncertainty of trajectory length and other factors, this thesis proposes a vehicle trajectory classification algorithm based on LSTM.

5.1 Classification of multiple trajectories based on LSTM

5.1.1 Algorithm selection basis

Because the time that the vehicle stays on the screen is uncertain, resulting in uncertainty in the length of its trajectory. Therefore, there are certain problems in classifying vehicle behavior through the total trajectory of the vehicle. At the same time, because the vehicle tracking is real-time, the vehicle tracking data that can be obtained at a certain moment does not include the vehicle tracking data at the previous moment. This requires the classification model to give an output immediately given an input, instead of having to accumulate a string of inputs to give an output, as shown in Figure 5.2. And the inputs of Figure 5.1 are not the real time inputs.

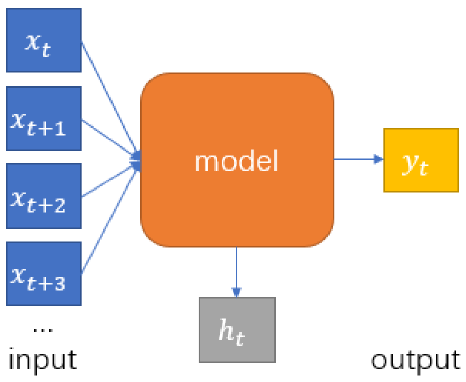


Fig 5.1 Track vehicle by delay input

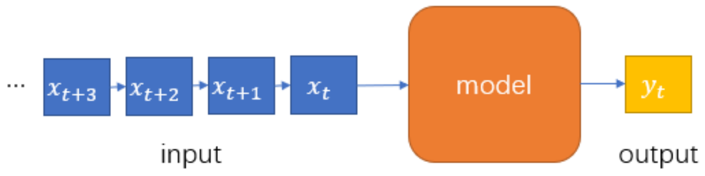


Fig 5.2 Track vehicle by real-time input

The behavior recognition of a vehicle is not only based on the location information of the vehicle at the current moment, but also on the location information of a series of previous moments. Therefore, the vehicle trajectory classification system is required to have a certain degree of memory, which can link the current information with the previous information for judgment, and LSTM can just meet the demand.

5.1.2 Memory characteristics of LSTM

The input of LSTM is composed of three parts, one is the data x_t that needs to be calculated, and the other two are h_{t-1} and c_{t-1} derived from the last time LSTM. The output of LSTM is h_t and c_t. The main structure of the network is shown in the Figure 5.3. Among them, h_t is the main output, which will be connected to the following algorithms, such as dropout layer, Full Connect layer etc.

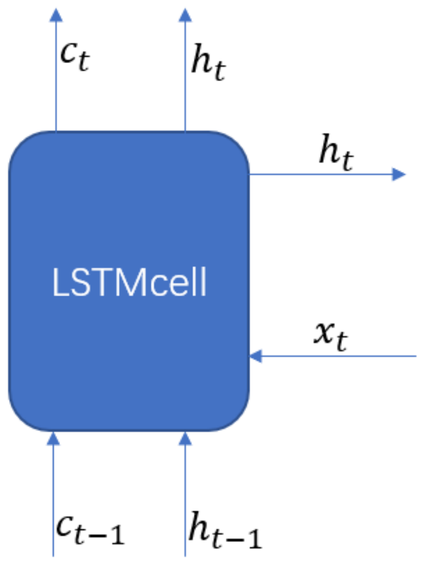


Fig 5.3 Main structure of LSTM

The internal structure of LSTM is shown in Figure 5.4. In the structure diagram, we can see that h_t and c_t are spliced first, and then matrix multiplication is performed with four weights. σ is the sigmoid activation function, which is used as a switch to control the memory behavior.

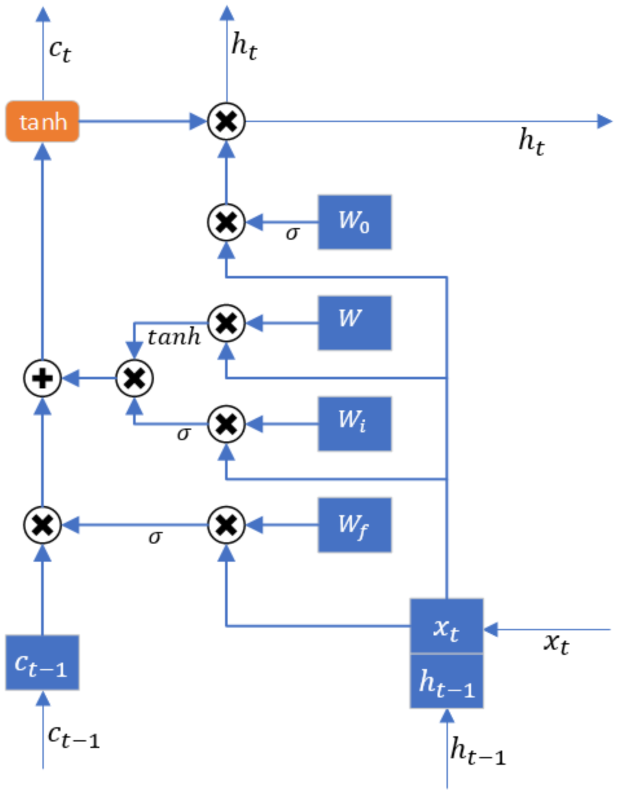


Fig 5.4 The internal structure of LSTM

The memory characteristics of LSTM are mainly related to h_{t-1} and c_{t-1}. The composition of its memory characteristics mainly has three parts:

The first part is to forget the past part, which is the path through W_f. This part is mainly to selectively discard the input that came in the previous moment. It will abandon the unimportant and leave the important.

The second part remembers the current part, which is the path through W_i. This part selectively memorizes the input at the current moment. Record what is important, and write less of what is not.

The third part is the control output part, which is the part that passes W_o. This part will determine which will be regarded as the output of the current state. It is mainly controlled by W_o and σ.

5.1.3 Format of input and output

Bounding box's *x*, *y*, *w*, *h*. For a certain moment, the only data that can be used for vehicle behavior recognition is the location information of the tracked vehicle. This information comes from the center coordinates of the bounding box of the tracked vehicle. At the same time, the aspect ratio of bounding Box is introduced. The aspect ratio is introduced because different orientations of the vehicle will cause the aspect ratio to change, which is an important basis for judging whether the vehicle is turning. Therefore, it was originally considered that the input data is a 3-dimensional vector (x, y, r), including the x coordinate, y coordinate, and aspect ratio r of the bounding box. But if we use the aspect ratio, there will be a problem, that is, the value of r may be greater than 1, which is not very good for training the model, so we finally decided to still use the x, y, w, h of the bounding box as the input.

The vehicle trajectory behavior classification is finally divided into 3 categories, namely parking behavior, leaving behavior, and no behavior. So, the final output is a 3-dimensional vector, and each dimension of the vector value is represented as the confidence of the category. Finally, the category with the highest confidence is taken as the classification result of this time. In order to achieve 3 classifications and output a 3-dimensional vector, a fully connected layer is added after the output of LSTM.

5.2 Multi-track real-time parallel classification

5.2.1 Real-time classification

Real-time classification refers to the classification of a piece of data immediately. Since the c and h vectors in the LSTM have memorized the trajectory characteristics before the current moment, when a new data comes, it can be classified immediately.

5.2.2 Parallel classification

At a certain moment or in a certain frame, there is not only one trajectory that needs to be classified. After vehicle tracking, if there are n vehicles, the trajectories of n vehicles need to be classified. This is a parallel classification process. The structure of parallel classification is shown in Figure 5.5.

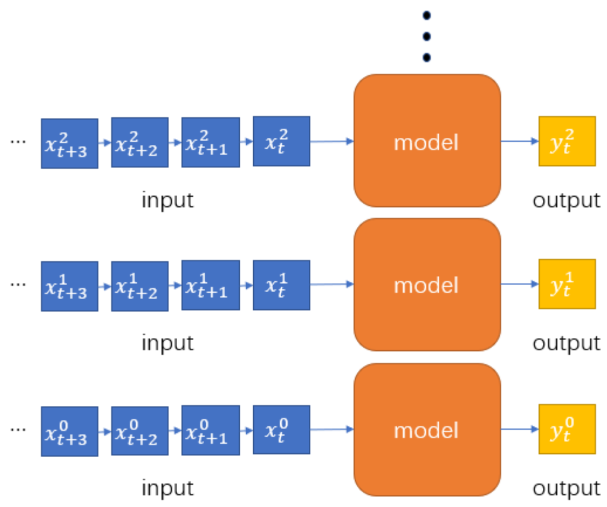


Fig 5.5 Parallel classification of multiple trajectories

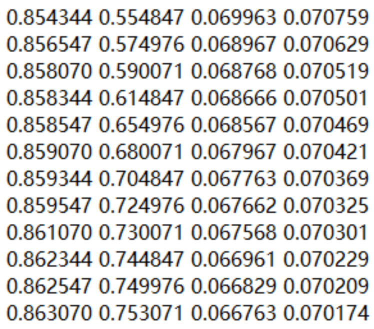
The superscript of x represents the track number of the tracked vehicle, and the subscript represents the time. Similarly, the superscript of y represents the serial number of the tracked vehicle, and the subscript represents the time. Here, only the output at time t is drawn.

Multiple trajectories in parallel classification are managed by a class named *RealTimeLSTM* customized in this project. This class contains two matrices, h and c, and the size is 100×4, where 100 refers to managing up to 100 trajectories at the same time, and 4 refers to the size of the hidden layer of LSTM. The program has two other parts. One is to add a trajectory. When a new trajectory is generated, the new trajectory needs to be added to the classified trajectory list and then classified. The second is to clear the trajectory, because the length of the trajectory list is fixed. When a trajectory has disappeared, it needs to release its occupation in the object generated by *RealTimeLSTM*, and reset the c and h parameters of the previous trajectory to avoid the classification result of the new added trajectory is affected.

5.3 Model training

5.3.1 Dataset introduction

The data set of this thesis contains a total of 700 trajectories, which are divided into two folders, one folder stores the training set, a total of 600 trajectory data, and one folder stores the validation set, a total of 100 trajectory data. Each folder contains 3 sub-folders, which respectively represent parking behavior, leaving behavior, and no behavior. Each sub-folder stores multiple files, and each file represents the driving trajectory of a car. Each line in each file has 4 data, which are the *x*, *y*, *w*, and *h* of the bounding box. The format is shown in Figure 5.6.



```
0.854344 0.554847 0.069963 0.070759
0.856547 0.574976 0.068967 0.070629
0.858070 0.590071 0.068768 0.070519
0.858344 0.614847 0.068666 0.070501
0.858547 0.654976 0.068567 0.070469
0.859070 0.680071 0.067967 0.070421
0.859344 0.704847 0.067763 0.070369
0.859547 0.724976 0.067662 0.070325
0.861070 0.730071 0.067568 0.070301
0.862344 0.744847 0.066961 0.070229
0.862547 0.749976 0.066829 0.070209
0.863070 0.753071 0.066763 0.070174
```

Fig 5.6 vehicle trajectory format

In the network model, the correspondence between the data in the trajectory dataset and the label is as follows:

When a file is in the folder representing the parking behavior, except for the last line of data corresponding to the label vector (*1, 0, 0*), the other lines of data corresponding to the label are all (*0, 0, 1*).

When a file is in the folder representing the leave behavior, except for the last row of data corresponding to the label vector (*0, 1, 0*), the labels corresponding to the data of the other rows are all (*0, 0, 1*).

When a file is in the folder representing no behavior, the label corresponding to each row of data is (*0, 0, 1*).

5.3.2 Experimental result

The model effect is shown in Table 5.1, where speed represents the time it takes for the model to calculate a row of data, the input data dimension is 6×4, 6 represents 6 vehicles, and 4 is the dimension of vector (*x, y, w, h*) of the bounding box. It can be seen that the time consumed by using this model to classify the trajectory is almost negligible, and the classification effect can also meet the requirements of this dataset.

Table 5.1 speed of RealTimeLSTM net

Model	GPU speed (ms)	CPU speed (ms)	Accuracy (%)
RealTimeLSTM Net	0.003	0.49	94.86

5.4 Summary

This chapter first introduces the problem that this thesis needs to solve is to realize the classification of trajectories, while also satisfying the real-time and parallelism of multiple trajectory classifications. Because of the uncertainty of the length of the vehicle trajectory, LSTM was finally selected as the basic algorithm for the trajectory classification solution. Then it introduces how to use the memory characteristics of LSTM to realize the classification of vehicle trajectories. Finally, we train the model, and then the evaluation results are given.

6 Implementation and result of RPBR system

6.1 Implementation of system

6.1.1 System overview

This system contains 4 modules in total, which are the Vehicle Detection and Recognition module introduced in detail in Chapter 2, the License Plate Recognition module introduced in detail in Chapter 3, the Vehicle Tracking module introduced in detail in Chapter 4, and the Vehicle Trajectory Classification module introduced in detail in Chapter 5. The relationship between the various modules is shown in Figure 6.1.

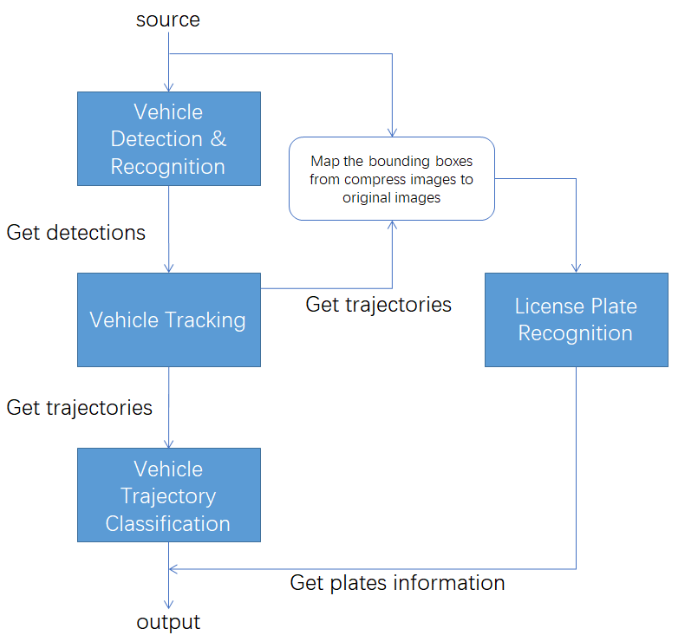
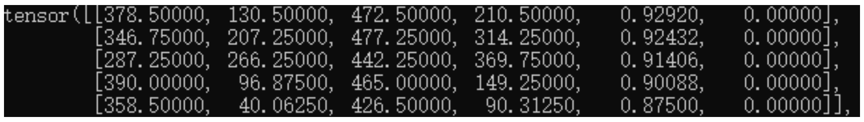

source
Vehicle Detection & Recognition
Map the bounding boxes from compress images to original images
Get detections
Vehicle Tracking
Get trajectories
License Plate Recognition
Get trajectories
Vehicle Trajectory Classification
Get plates information
output

Fig 6.1 Relationship between the modules

6.1.2 Get detections to vehicle tracking

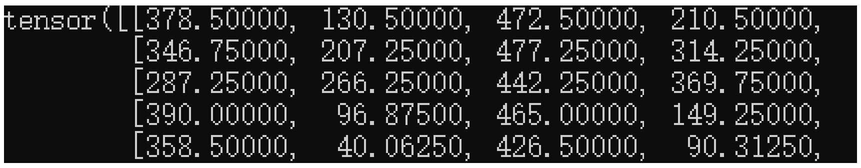
After the Vehicle Detection and Recognition module, we got the detections. Detections are the information of the identified vehicles, which include location information, category information, confidence information, and so on. Its shape is $n \times 6$, where n represents the number of vehicles contained in this frame of image. 6 represents 6 dimensions of information. The first dimension to the fourth dimension is represent the position information of the vehicle, and its format is (min *x*, min *y*, max *x*, max *y*). The fifth dimension represents confidence. The sixth dimension represents category. The category in this system is only car, so the value of the sixth dimension is always 0. We take out a data of detections from the test data as an example. The detections contain 5 vehicles. The data is shown in Figure 6.2.



```
tensor([[378.50000, 130.50000, 472.50000, 210.50000,    0.92920,    0.00000],
        [346.75000, 207.25000, 477.25000, 314.25000,    0.92432,    0.00000],
        [287.25000, 266.25000, 442.25000, 369.75000,    0.91406,    0.00000],
        [390.00000,  96.87500, 465.00000, 149.25000,    0.90088,    0.00000],
        [358.50000,  40.06250, 426.50000,  90.31250,    0.87500,    0.00000]],
```

Fig 6.2 format of detections

Then we need to split the detections, we only need the data from the 1st dimension to the 4th dimension, as shown in Figure 6.3.



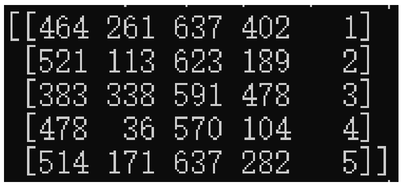
```
tensor([[378.50000, 130.50000, 472.50000, 210.50000,
        [346.75000, 207.25000, 477.25000, 314.25000,
        [287.25000, 266.25000, 442.25000, 369.75000,
        [390.00000,  96.87500, 465.00000, 149.25000,
        [358.50000,  40.06250, 426.50000,  90.31250,
```

Fig 6.3 split of detections

Then use the cut detections and the image resized by the Vehicle Detection and Recognition module as the input of the Vehicle Tracking module.

6.1.3 Get trajectories to vehicle trajectory classification

After the Vehicle Tracking module, we will get a $m \times 5$ data. Where m represents the number of tracks, and m in tracks is greater than or equal to n in detections. 5 represents information of 5 dimensions. The first to fourth dimensions represent the position information of the bounding boxes, in the format (min *x*, min *y*, max *x*, max *y*), and the fifth dimension represents the id of the track. Its data example is shown in Figure 6.4.



```
[[464 261 637 402   1]
 [521 113 623 189   2]
 [383 338 591 478   3]
 [478  36 570 104   4]
 [514 171 637 282   5]]
```

Fig 6.4 format of tracks

Then take out the data from the 1st dimension to the 4th dimension and convert it to

YOLO format (*x, y, w, h*), the conversion formula is:

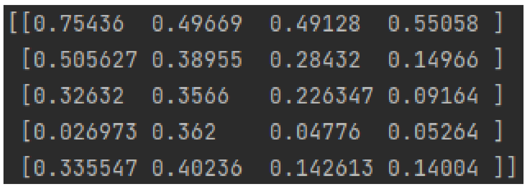
$$x = \frac{\min x + \max x}{2 \times imageWidth} \tag{6.1}$$

$$y = \frac{\min y + \max y}{2 \times imageHeight} \tag{6.2}$$

$$w = \frac{\max x - \min x}{imageWidth} \tag{6.3}$$

$$h = \frac{\max y - \min y}{imageHeight} \tag{6.4}$$

An example of the converted data is shown in Figure 6.5.



```
[[0.75436  0.49669  0.49128  0.55058 ]
 [0.505627 0.38955  0.28432  0.14966 ]
 [0.32632  0.3566   0.226347 0.09164 ]
 [0.026973 0.362    0.04776  0.05264 ]
 [0.335547 0.40236  0.142613 0.14004 ]]
```

Fig 6.5 data format of *xywh*

After obtaining such data, the data and the separated id are used as the input of Vehicle Trajectory Classification module.

6.1.4 Get trajectories to license plate recognition

As we have said before, after the Vehicle Tracking module, we will get $m \times 5$ data. Then map the bounding boxes from compressed image sizes to source image sizes. After the new data is obtained, the source images are cropped using this new data. Through the comparison in Figure 6.6, we can see that the license plate information in the original image is clearer than the license plate information in the compressed image. The resolution of the license plate recognizes the license plate. The accuracy of this step has a great influence, so this step is very important.

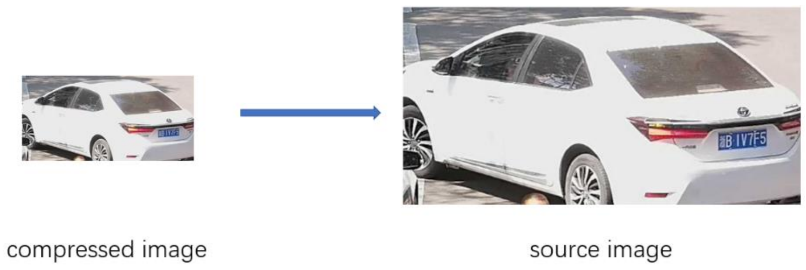


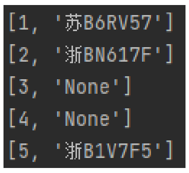
Fig 6.6 map the vehicle from compressed image to source image

Finally, integrate these images together. Combine these images with the id in each track as the input of the License Plate Recognition module.

6.1.5 plates information and result of classification

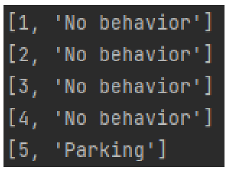
After the License Plate Recognition module, we can get the license plates number information and their corresponding ids, as shown in Figure 6.7.

After the Vehicle Trajectory Classification module, we can get the result of trajectories classification and their corresponding ids, as shown in Figure 6.8.



```
[1, '苏B6RV57']
[2, '浙BN617F']
[3, 'None']
[4, 'None']
[5, '浙B1V7F5']
```

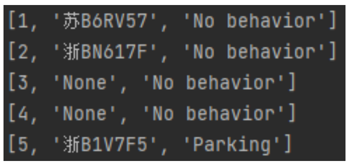
Fig 6.7 id and license plates information



```
[1, 'No behavior']
[2, 'No behavior']
[3, 'No behavior']
[4, 'No behavior']
[5, 'Parking']
```

Fig 6.8 id and behavior classification

Combine the output of the License Plate Recognition module with the output of the Vehicle Trajectory Classification module through id, and link the trajectory classification results with the license plate number information, as shown in Figure 6.9.



```
[1, '苏B6RV57', 'No behavior']
[2, '浙BN617F', 'No behavior']
[3, 'None', 'No behavior']
[4, 'None', 'No behavior']
[5, '浙B1V7F5', 'Parking']
```

Fig 6.9 The union of license plate information and behavior classification

In this way, we successfully bind parking behaviors to license plates. We can record the parking behaviors of the vehicles through the license plates, and then charge the owners based on the parking behavior records.

6.2 Function description

6.2.1 System startup configuration

When the system is started, you can customize configure the system to ensure the system run normally and facilitate the debugging of the system. The custom commands and their function descriptions are shown in Table 6.1.

Table 6.1 the custom commands and function descriptions

command	description
--detect_weights	The weights of vehicle detection and recognition model
--plate_weights	The weights of license plate recognition model
--track_weights	The weights of HVS model
--source	The input source of the system, can choose video from RSTP
--device	Choose CPU or GPU

6.2.2 Output format

The output of the system is displayed in real time. Each vehicle is marked by a red, blue or green bounding box according to its behavior status, and its license plate characters and current behavior status are displayed on the bounding box, as shown in Figure 6.10.

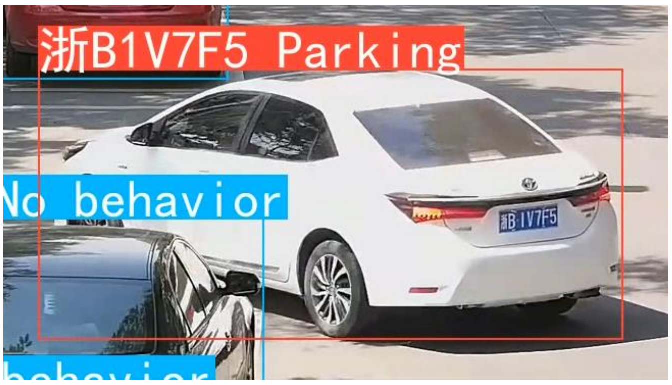


Fig 6.10 example of output format

When the license plate characters of the vehicle are recognized, the license plate number will be displayed. If it is not recognized, it will display *none*.

When the vehicle is in the parking state, the label above the bounding box will display

Parking, and the color of the bounding box is red. When the vehicle is in leaving state, the label above the bounding box will display *Leaving*, and the color of the bounding box will be green. When the vehicle is in the *no behavior* state, the label above the bounding box displays *no behavior*, and the color of the bounding box is blue.

6.3 System testing

6.3.1 Speed testing

The time required for this system to process one frame of image under Nvidia Geforce 2060 and Intel core i5-10400 is shown in Table 6.2.

Table 6.2 result of speed testing

Numbers of vehicles	GPU speed (s)	CPU speed (s)
5	0.040	0.110

When the system uses GPU for computing, it can meet real-time requirements, but it is a bit difficult under CPU.

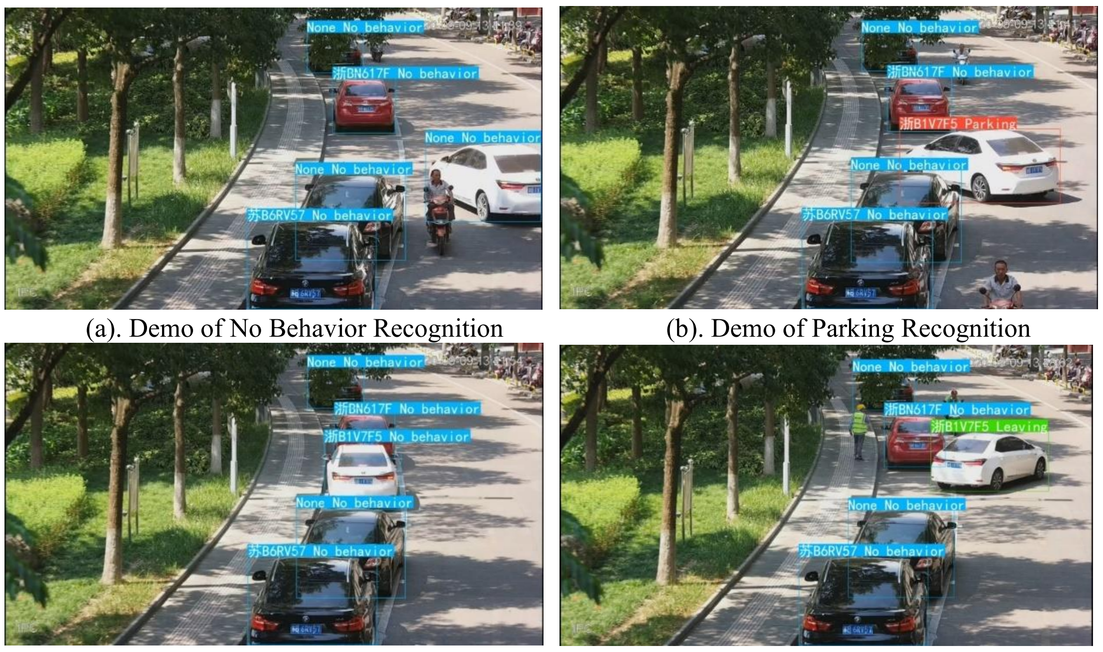
6.3.2 Function testing

In Figure 6.11(a), because the license plate of the white vehicle is not completely displayed, the license plate characters cannot pass the verification code, so the area above the bounding box where the license plate characters should be displayed is *none*. Also, because the current vehicle behavior is identified as *no behavior*, the behavior status display area displays *no behavior*, and the bounding box is blue. The vehicle covered by leaves in Figure 6.11 can still be recognized even if the covered area is large. Because its license plate is covered and cannot display completely, it cannot be correctly recognized, so it is marked as *none*.

In Figure 6.11(b), the license plate information of the white vehicle is accurately identified and displayed above the bounding box. Since the system detects that the white vehicle is parking and marks its behavior status as Parking, the behavior display area shows *parking* and the bounding box is red.

In Figure 6.11(c), the behavior of the white vehicle is recognized as *no behavior* by the system, so the behavior display area shows *no behavior*, and the bounding box is blue. At the same time, it can be seen that although the license plate of the red vehicle in the picture is obscured by the white vehicle, the license plate can still be displayed accurately, which shows that even if the vehicle is partially obscured, the system can still track it.

In Figure 6.11(d), the license plate information of the white vehicle is accurately identified and displayed above the bounding box. The current behavior of the vehicle leaving the parking space is correctly identified as *leaving*, the behavior display area shows *leaving*, and the bounding box is green.



(a). Demo of No Behavior Recognition (b). Demo of Parking Recognition

(c). Demo of Plate Marked (d). Demo of Leaving Recognition

Fig 6.11 Demonstration

6.4 Summary

After testing, it can be seen that the system can accurately detect and recognize the vehicle, even if some parts of the vehicle are covered, it can be correctly detected and recognized.

The system can correctly recognize the characters of the license plate when the license plate is displayed clearly. The license plate verification algorithm can filter out incomplete license plate information when the license plate characters are displayed incomplete, and it can also filter out the license plate information when the license plate information is low in confidence, so as not to record wrong license plate information and cause wrong parking bills.

The system can continuously track the vehicle. When the vehicle is partially covered even if the license plate is also covered, the license plate information of the vehicle can still be displayed at this time. This means that the license plate information displayed at this time comes from the record when the license plate is not covered, indicating the vehicle is being tracked continuously and has not been lost by the tracker.

The system can accurately classify the behavior of the vehicle and bind its behavior to the license plate. And when the system uses GPU for computing, it can meet real-time requirements, but it is a little difficult on the CPU.

7 Conclusion and future work

7.1 Conclusion

Based on the current shortage of urban parking space and the high cost of roadside parking management, this thesis proposes an image recognition method based on monocular cameras and deep learning to classify the behavior of vehicles on the roadside.

Based on the YOLOv5 algorithm, this paper realizes the detection and recognition of vehicles and then proposes a trapezoidal convolution algorithm. Compared with the traditional convolution algorithm, the trapezoidal convolution algorithm has a great improvement in speed. After replacing the convolution algorithm, when the new model is applied to the scene of roadside parking behavior recognition, the accuracy of the algorithm does not significantly decrease. Then, based on the current status of the research field of license plate recognition, this thesis proposes a one-stage license plate recognition algorithm based on the YOLOv5 algorithm, which greatly simplifies the complex process of the traditional license plate recognition field. Regarding vehicle tracking, this paper proposes the HVS model, which can effectively extract the features of the vehicle, and the model is fast, can improve the efficiency of vehicle tracking, has a small number of parameters, and can reduce memory usage. Then, based on the memory characteristics of LSTM, this paper proposes implementing vehicle trajectory classification through the LSTM network model, and the final classification effect is very good.

This thesis combines these modules to form a complete RPBR system. Then various functions of the system are tested, and after continuous debugging and improvement, the system has been gradually improved.

7.2 Future work

This thesis improves some existing technologies and integrates some of our ideas into the implementation of the system, which makes the system more efficient and complete. But there are still many areas that need to be strengthened. The first thing that needs to be strengthened is the training of the model. The number of data sets used for training in this paper is not large, which causes the problem that the generalization ability of the model may be relatively poor. In addition, the test scenarios in this article are limited, and some extreme scenarios, such as rainy days and nights, are not tested. But in theory, as long as the data set is rich enough, the generalization ability of the trained model will be stronger. Therefore, if conditions permit, more abundant data sets will be collected in the future, and the RPBR system will be tested and improved in more complex scenarios to make it more compatible and have a wider range of applications. The second is that this thesis can continue to be optimized for the CPU. At present, the time spent by the RPBR system on the CPU is about 0.1s. If the architecture of the system is improved, such as using multi-threading, running the Vehicle Detection and Recognition Module and License Plate Recognition module with one thread each can improve the efficiency of the entire system.

Reference

[1] YU Qing-qing, ZHANG Ke, TANG Heng-liang, et al. A Roadside Parking Detection and Recognition Method Based on Video Motion Trail[J]. Computer and Modernization, 2017, (9):67-73.

[2] Shi J, Tomasi C. Good features to track[C] Proceedings of the 7th IEEE Computer Society Conference on Computer Vision and Pattern Recognition. 1994: 593-600.

[3] David V, Sanchez A. Advanced support vector machines and kernel methods[J]. Neurocomputing, 2003, 55(1):5-20.

[4] Hu X, Huang J, Yuan M. Real-time Detection of Roadside Parking Events Based on Video[J]. Information & Communications, 2019.

[5] Girshick R, Donahue J, Darrell T, et al. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation[C] IEEE Computer Society. IEEE Computer Society, 2013.

[6] Girshick R. Fast R-CNN[J]. Computer Science, 2015.

[7] Ren Shaoqing et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks[J]. IEEE transactions on pattern analysis and machine intelligence, 2017, 39(6): 1137-1149.

[8] Dai J, Li Y, He K, et al. R-FCN: Object Detection via Region-based Fully Convolutional Networks[J]. Curran Associates Inc. 2016.

[9] Liu W, Anguelov D, Erhan D, et al. SSD: Single Shot MultiBox Detector[J]. Springer, Cham, 2016.

[10] Everingham M, Gool L V, Williams C K I, et al. The Pascal Visual Object Classes (VOC) Challenge[J]. International Journal of Computer Vision, 2010, 88(2):303-338.

[11] PEI Qiao-na. Moving Objects Detection and Tracking Techniques based Optical Flow[D]. North China University of Technology.

[12] Arulampalam M S, Maskell S, Gordon N, et al. A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking[J]. IEEE Transactions on Signal Processing, 2002, 50(2): 174-188.

[13] D Comaniciu, Meer P. Mean shift: a robust approach toward feature space analysis[J]. IEEE Trans Pattern Analysis & Machine Intelligence, 2002, 24(5): 603-619.

[14] Kim K I, Jung K, Jin H K. Texture-based approach for text detection in images using support vector machines and continuously adaptive mean shift algorithm[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2003, 25(12): 1631-1639.

[15] Henriques J F, Caseiro R, Martins P, et al. High-Speed Tracking with Kernelized Correlation Filters[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2015, 37(3): 583-596.

[16] Danelljan M, Hager G, Khan F S, et al. Learning Spatially Regularized Correlation Filters for Visual Tracking[C] 2015 IEEE International Conference on Computer Vision (ICCV). IEEE, 2015.

[17] Zhu G, Porikli F, Li H. Beyond Local Search: Tracking Objects Everywhere with Instance-Specific Proposals[J]. IEEE, 2016.

[18] Babenko B, Yang M H, Belongie S. Visual tracking with online Multiple Instance Learning[C] Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. IEEE, 2009.

[19] Oza N C, Russell S. Online Bagging and Boosting[C] IEEE International Conference on Systems. IEEE,

2006.

[20] Hare S, Golodetz S, Saffari A, et al. Struck: Structured Output Tracking with Kernels[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2015:2096-2109.

[21] Zhang J, Ma S, Sclaroff S. MEEM: Robust Tracking via Multiple Experts Using Entropy Minimization[C] European Conference on Computer Vision. Springer International Publishing, 2014.

[22] Kalal Z, Mikolajczyk K, Matas J. Face-TLD: Tracking-Learning-Detection applied to faces[C] IEEE International Conference on Image Processing. IEEE, 2010.

[23] Bolme D S, Beveridge J R, Draper B A, et al. Visual object tracking using adaptive correlation filters[C] The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010. IEEE, 2010.

[24] Henriques J F, Rui C, Martins P, et al. Exploiting the Circulant Structure of Tracking-by-Detection with Kernels[C] Proceedings of the 12th European conference on Computer Vision - Volume Part IV. Springer, Berlin, Heidelberg, 2012.

[25] Galoogahi H K, Fagg A, Lucey S. Learning Background-Aware Correlation Filters for Visual Tracking[J]. 2017 IEEE International Conference on Computer Vision (ICCV), 2017.

[26] Yang L, Zhu J. A Scale Adaptive Kernel Correlation Filter Tracker with Feature Integration[C] European Conference on Computer Vision. Springer, Cham, 2014.

[27] Bewley A , Ge Z, Ott L, et al. Simple Online and Realtime Tracking[C] 2016 IEEE International Conference on Image Processing (ICIP). IEEE, 2016.

[28] Blackman, S. Multiple hypothesis tracking for multiple target tracking[J]. IEEE Aerospace & Electronic Systems Magazine, 2009, 19(1):5-18.

[29] Habtemariam B, Tharmarasa R, Thayaparan T, et al. A Multiple-Detection Joint Probabilistic Data Association Filter[J]. IEEE Journal of Selected Topics in Signal Processing, 2013, 7(3):461-471.

[30] Wojke N, Bewley A, Paulus D. Simple Online and Realtime Tracking with a Deep Association Metric[J]. IEEE, 2017:3645-3649.

[31] Silva S M, Jung C R. License Plate Detection and Recognition in Unconstrained Scenarios[C] 2018.

[32] Hochreiter S, Schmidhuber J. Long Short-Term Memory[J]. Neural Computation, 1997, 9(8):1735-1780.

[33] Redmon J, Divvala S, Girshick R, et al. You Only Look Once: Unified, Real-Time Object Detection[C] Computer Vision & Pattern Recognition. IEEE, 2016.

[34] Redmon J, Farhadi A.Yolo9000：better, faster, stronger[C] Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017：6517-6525.

[35] Redmon J, Farhadi A. YOLOv3: An Incremental Improvement[J]. arXiv e-prints, 2018.

[36] LIU Jing-yu, LIU De-er, YANG Peng, et al. License Plate recognition with partial occlusion based on CNN [J]. Measurement & Control Technology, 2021, 40(2): 53-57, 63. DOI: 10.19708/j.ckjs. 2021.02.010.

[37] BI Bo, SHAO Yong-qian, SUN Dong-jun, et al. License plate recognition based on OpenCV [J]. Electronic Design Engineering,2019,27(1): 37-41.

[38] WU An-hui, HE Jia-feng, HE Qi-li. Research on Algorithm of License Plate Detection and Recognition in Complex Scenes [J]. Modern Information Technology,2021,5(1):81-83,87.

[39] GUO Feng, WANG Bing-zheng, CHEN yan. Vehicle tracking algorithm in road traffic video[J]. Journal

of Zhengzhou University of Light Industry (Natural Science Edition), 2012(03):74-76.

[40] ZHAO Qian-qian, SONG Huan-sheng, XU Xiao-juan, et al. Trajectory Extraction in 3D Space from Low Camera[J]. Video Engineering, 2015, 39(003):115-118.

[41] GUO Feng, WANG Bing-zheng, YANG Chen-hui. The Improved Algorithm for Vehicle Tracking and Retrograde Motion Detection under the Complicated Background [J]. Journal of Graphics, 2013, 34(004):150-153.

[42] ZHAO Sheng, ZHAO Xue-jian, ZHANG Xin-hui, et al. Vehicle Trajectory Recognition System Based on Sobel Operator and CNN [J]. Computer Technology and Development, 2018, 028(007):169-172.

[43] ZHANG Jian-ming, JIN Xiao-kang, WU Hong-lin, et al. Multi-model Real-time Compressive Tracking [J]. Journal of Electronics & Information Technology, 2018, 40(10):92-99.

[44] WANG Xiao-jia, ZENG Ying-kun. Study on Vehicle Travel Path Extraction Based on Close Range Photogrammetry [J]. Science Technology and Engineering, 2012, 12(020):5105-5108.

[45] DU Yun-ming, GAI Li-na, TIAN Jing. Vehicle Tracking Algorithm based on Sequential Monte Carlo Filter[J]. INTELLIGENT COMPUTER AND APPLICATIONS, 2012(05):4-7.

[46] LYU Yun-qiu, LIU Kai, FEI Ju-feng, et al. Real-time Tracking Method Based on Compressive Tracking and Genetic Algorithm[J]. Guidance & Fuze,2016,37(4):34-39,53. DOI:10.3969/j.issn.1671-0576.2016.04.007.

[47] LI Mu-zi. Design and Implementation of Automatic Detection and Tracking System based on Image Recognition[D]. Beijing University of Post and Telecommunications, 2019.

[48] Henriques J F, Caseiro R, Martins P, et al. High-Speed Tracking with Kernelized Correlation Filters[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2015, 37(3):583-596.

[49] Chen L, Ai H, Zhuang Z, et al. Real-Time Multiple People Tracking with Deeply Learned Candidate Selection and Person Re-Identification[C] IEEE International Conference on Multimedia & Expo. IEEE Computer Society, 2018:1-6.

[50] Liu X, Wu L, Tao M, et al. A Deep Learning-Based Approach to Progressive Vehicle Re-identification for Urban Surveillance[C] Springer International Publishing. Springer International Publishing, 2016.

[51] Zhou Y, Liu L, Shao L. Vehicle Re-Identification by Deep Hidden Multi-View Inference[J]. IEEE Transactions on Image Processing, 2018, PP(7):1-1.

[52] Yang B, Wang Y Z. Improved Vehicle Re-Identification Algorithm Based on DRDL Model[J]. Modern Computer, 2019.

[53] Peng J, Wang H, Zhao T, et al. Learning multi-region features for vehicle re-identification with context-based ranking method[J]. Neurocomputing, 2019, 359(SEP.24): 427-437.

[54] Liu X, Liu W, Mei T, et al. PROVID: Progressive and Multimodal Vehicle Reidentification for Large-Scale Urban Surveillance[J]. IEEE Transactions on Multimedia, 2018: 1-1.