CS405 proposal - Traffic Sign Detection and Recognition

A Proposal Prepared for the
Final Project of the graduate Course
CS405: Machine Learning

Prepared by:
Yaqi Wang, Jiajian Yang, Duolei Wang, Guang Chen
{12012936, 12012711, 12012727, 12032433}@mail.sustech.edu.cn
Department of Computer Science and Engineering
Southern University of Science and Technology

ABSTRACT

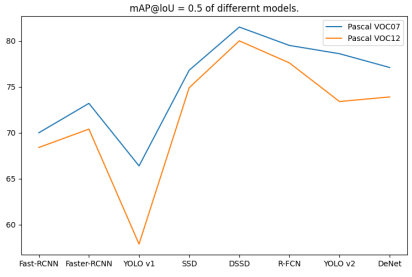
Traffic Sign Detection and Recognition (TSDR) technique is an important part of self-driving technology and a critical step for ensuring vehicle safety. For the final project, a YOLOv5 based model and data augment methods are used for real-time detection and recognition of traffic sign. TT100K (Tsinghua-Tencent 100K) is chosen to be the training and test data set. Meanwhile, model will be tested on a video record of traffic conditions around SUSTech. Other advanced goals including cars detection and tracking, distance estimation and lane detection have also been implemented.

2.1 Research Question or Problem

In recent years, with the rapid development of the economy, the number of vehicles on the traffic road has gradually increased, which has caused frequent traffic accidents, and intelligent technology is conducive to solving this problem, and the traffic sign recognition system has emerged as the times require. Through the accurate detection and identification of traffic signs on the road, TSDR (Traffic Sign Detection and Recognition) can effectively remind drivers to regulate and drive safely, and reduce the occurrence of traffic accidents. Therefore, it is meaningful for us to explore how to detect and recognize traffic sign efficiently and accurately.

In real world, the effect of TSDR is greatly affected by lighting and occlusion. At the same time, autonomous driving technology needs to calculate road conditions in real time, and how to design algorithms to achieve efficient TSDR system is of great importance. With the help of computer vision algorithm, region-based methods extract possible region then put them into specify classifier to get the result. However, the low efficiency makes it impractical in real world cases. In order to balance the speed and accuracy of the algorithm, YOLO, and also other one-step model, is a better choice. In order to adapt the model to the real situation, we add additional processing on dataset, such as erasing, occlusion, and use the augmented dataset for training.

(a) Traffic sign with different lighting and occlusion



(b) Comparison of different model in VOC datasets

FIGURE 1. Expected visualization of advanced goal

2.2 Research Goals and Objectives

Basic Goal: Traffic Sign Detection and Recognition

1. Train the model to detect and recoginize traffic sign using TT100K dataset, and test this model on the test dataset.
2. Use data augment methods to produce more training images, and train the model again; Compare the test result before and after data augment methods are used.
3. Record videos of traffic conditions around SUSTech, test the performance of model and optimize it.

Advanced Goals: Cars Detection, Distance Estimation and Lane Detection

1. Detect cars and estimate the distance between cars and camera.
2. Detect lanes in the recorded video.

An expected result of our model is shown in Figure.

2.3 Research Design and Methods

You only look once (YOLO)

1. Introduction: YOLO is an advanced, real-time object detection system. It has been designed to be super easy to get started and simple to learn. Figure 1 demonstrates that YOLOv5 shows a better performance on COCO dataset comparing with EfficientDet [1].
2. Net Structure: we learned about the original program of YOLOv5, and find that the net structure of YOLOv5 can be divided into two parts: the backbone and the head. Figure 2 shows the process of the data. In the project, there is a yaml file describing the detailed structure of the training net, which is shown in figure 3. The Backbone part is the module of extracting features. It extract small, middle and large features respectively in different layers and pass them to the Head part. Then in the Head part, the Concat modules mix the features together. Finally, the model detects the target in three different scales and gives the prediction. In some extent, the whole model is quite similar to FPN (FeaturePyramidNetworks) (Figure 4) [2].

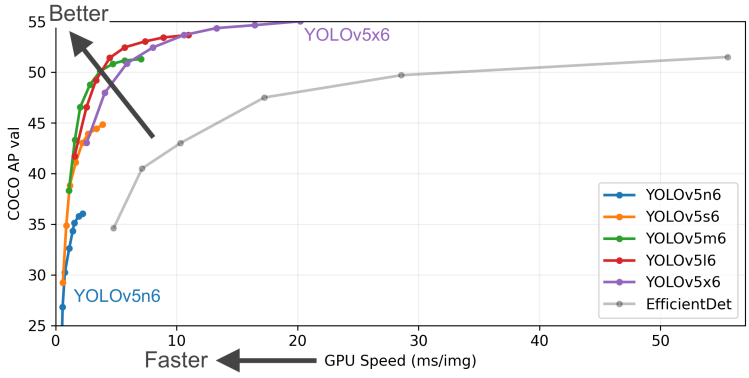


FIGURE 2. Test on COCO dataset

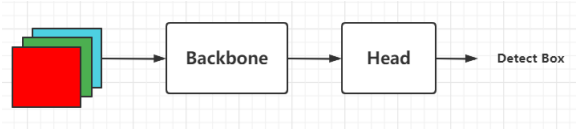


FIGURE 3. Main structure of YOLOv5

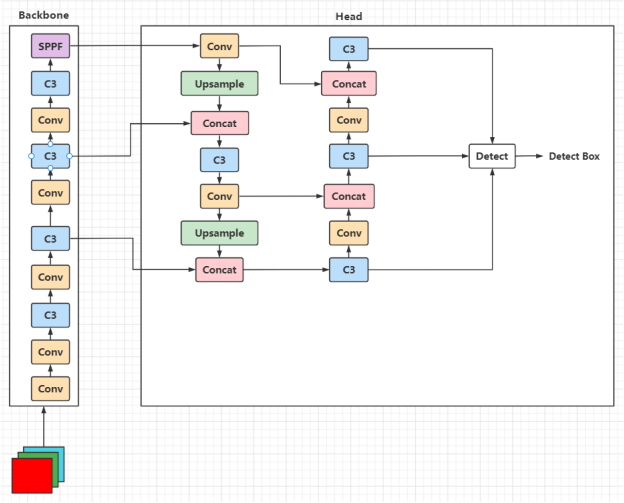


FIGURE 4. Main structure of YOLOv5

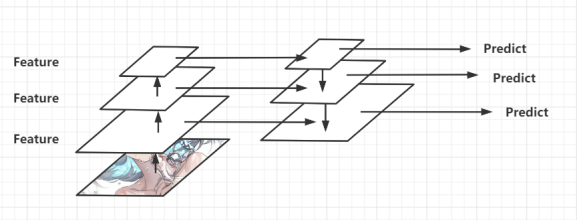


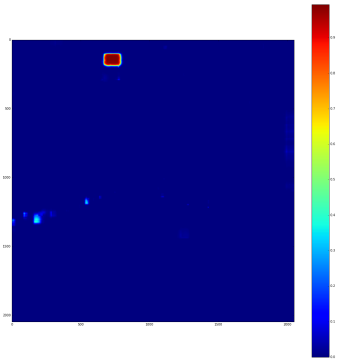
FIGURE 5. Model of FPN

Dataset and preprocessing

TT100K(Tsinghua-Tencent 100K) is a dataset which contains 100000 Tencent Street View panoramas, including large variations in illuminance and weather conditions in China. All of the traffic-sign in the pictures is annotated in a json-format file with a special class label, which marks the center of a box containing traffic-signs and also the normalized width and height. But the annotation used in YOLOv5 is the exact coordinate of the corners of the box. Thus we need to transform the label firstly using the formula, take the top left corner as example,

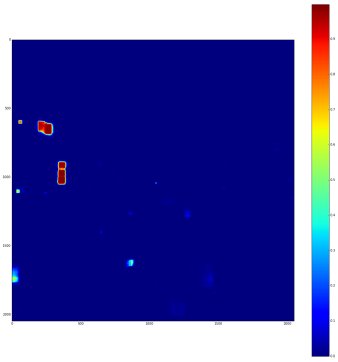
$$P_{Top,Up} = Center - \frac{1}{2} * Width - \frac{1}{2} * Height$$

(a) Sample 1



(b) Annotation 1

(c) Sample 2



(d) Annotation 2

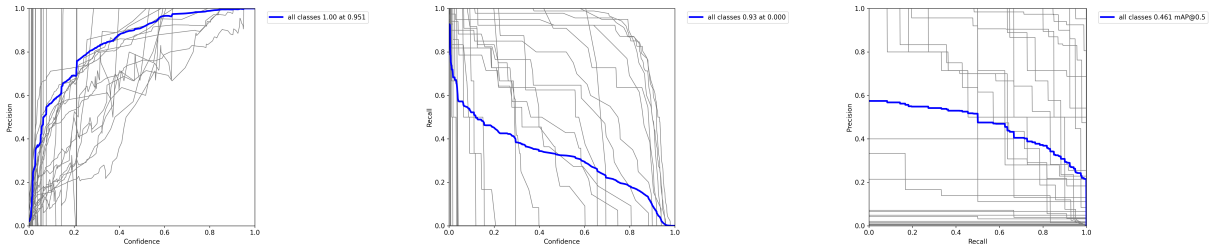
FIGURE 6. Examples of pictures and annotations

2.4 Traffic Sign Detection

A baseline YOLOv5 based model from github [3] is down load and implemented. We trained the baseline model using a small data set (1144images_dataset provided by Aryan Garg [4], 943 MB) for 50 epochs, and we tested the model on photos containing several traffic signs. Also, we tested this model on a video downloaded from website.

2.4.1 Model training

The training results are as following. We trained model with pretrained *yolov*5*s* (down load from github [5]) in that small data set for 50 epochs, the final $mAP@0.5$ (mean Average Precision at $IoU = 0.5$) achieves 0.46178, and $mAP@0.5:0.95$ (Average mAP at different IoU thresholds, from 0.5 to 0.95 with a step size of 0.05) achieves 0.31871.



(a) Precision Curve (b) Recall Curve (c) PR Curve

FIGURE 7. P,R,PR curves

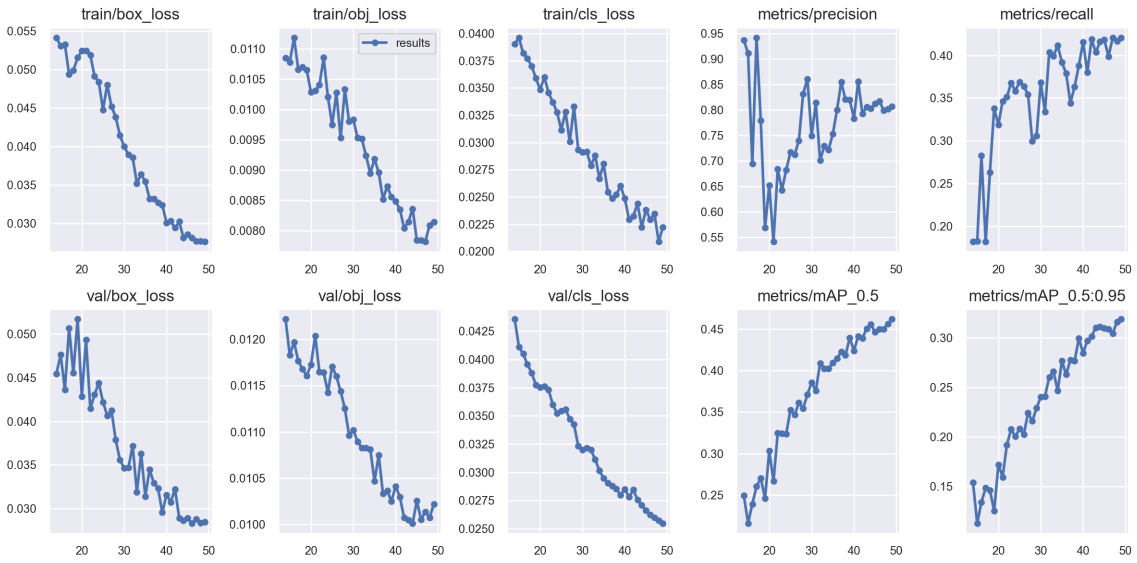


FIGURE 8. Training Results

2.4.2 Model testing and problem

We also test the baseline model on several screenshot photos. The original test picture 1 shown in Figure 8(a) is a screenshot taken from the main page of Chinese Traffic Sign Database [6]. (b) is a mosaic picture of several traffic signs. (c) is a mosaic picture of traffic signs occluded or photographed in adverse light conditions, and (d) is a simple test picture. Our base line model detect all the signs in (a) and the results were shown in (e), while there are many mistaken classifications, and one target may be detected with two (or more) boxes. As for (b), our model performs relatively well (results shown in (f)), however, there was a sharply oblique, large-scale sign not being successfully detected. In picture (g), none of these signs photographed in adverse conditions have been detected. As for the simple test picture, the single traffic sign was detected successfully in (h).

Overall, the drawbacks and possible reasons are

- Some of the signs of certain types were not detected (unsatisfactory recall), which may be partly attributed to the small-scale training set with possible incomplete variety of traffic signs.
- Fail to detect signs photographed in adverse conditions (strong lighting, obsure, occlusion, deformation and etc.), which may due to the lack of data augment procedure of the training data.
- False classification (unsatisfactory precision).

(a) original test picture 1 (b) original test picture 2 (c) original test picture with occlusion (d) original simple test picture

(e) detected test picture 1 (f) detected test picture 2 (g) detected test picture with occlusion (h) detected simple test picture

FIGURE 9. original and detected test pictures

For further improvement, we list the possible optimization methods to conquer the above drawbacks of model.

- Use data augmentation methods (like light transformation, shape transformation, sheering, random erasing, mosaic and etc.) to produce more training images with hard-to-recognize traffic signs.
- Train the model with a larger data set with complete traffic sign types and data produced by data augmentation.

2.4.3 Data augmentation

In this paragraph, we introduce the data augmentation methods used in our works. Data augmentation are a series of operations on the graph data. With various methods to deal with the origin graphs, new graphs are generate to improve the complex of train data. It can reduce the model overfitting because of the additional data. On the other hand, we can construct corresponding image samples through data augmentation for the above problems. In theory, data augmentation optimize the performance of YOLOv5 for these specific problems. In YOLOv5, researchers usually use Mosic, copy-paste, RandomPerspective, MixUp, Augment-HSV and Random-affine to expand data. There are three kinds of them we used:

- Mosic: Overlay multiple images, adjust their size and transparency to form a new image. This is equivalent to using the features of multiple samples to form a new sample and retain the differences between the data. At the same time, the new samples imitate occluded traffic signs, which may improve the performance of the model on this type of problem.

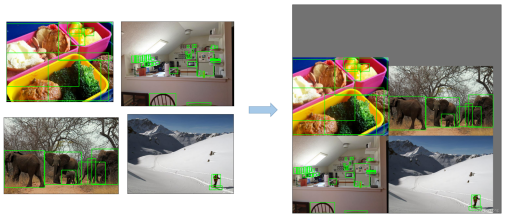


FIGURE 10. Mosic Results

- Augment-HSV: Increase or decrease the pixel brightness of certain areas in the original image, enhance or reduce their contrast, sharpness, etc. This method effectively constructs similar new images, similar to images taken under poor lighting conditions.

FIGURE 11. Augment-HSV Results

- Random-affine: Flip, symmetry, or crop the image to generate new data, which can effectively avoid over-fitting of the model.

FIGURE 12. Random-affine Results

Examples of augmented results for training the weights of the YOLOv5 model are shown below, which are from the TT100k dataset.

FIGURE 13. Augmented Results

Experiment

We used a server with nvida3090 graphics card to train the YOLOv5 model. The software version during training is: Python 3.8.2, pytorch 1.6.0 and CUDA 11.2. Based on the above content, we did some training experiments and compared the effect of data enhancement with 100 epochs. Finally, the training result of 300epochs is used as the weight of our traffic sign model.

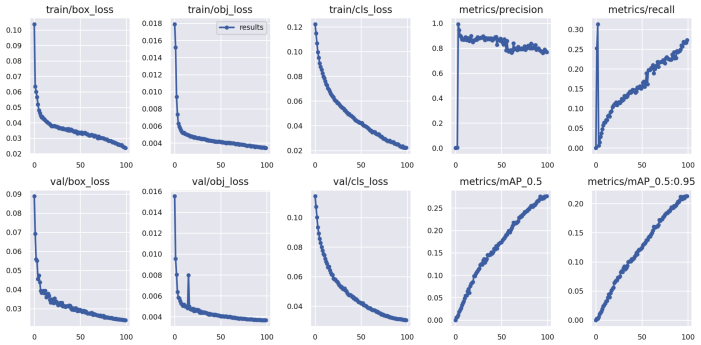


FIGURE 14. Training Performance without Augmentation

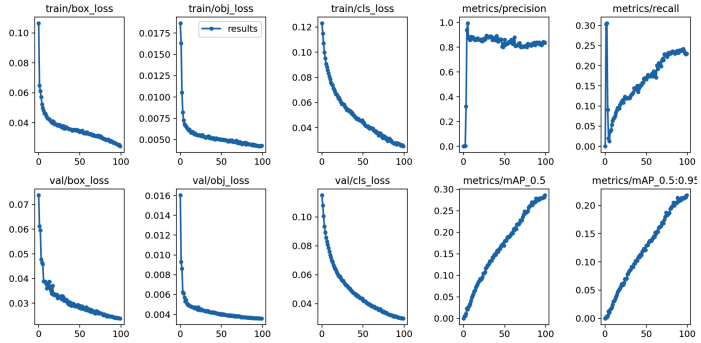


FIGURE 15. Training Performance with Augmentation

The training convergence speed after data augmentation is slowed down, but the test effect is better. Here we use a video taken from the campus road section of Southern University of Science and Technology as a test:

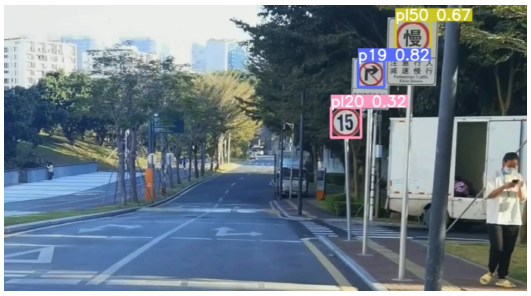


FIGURE 16. Test Results

2.5 Lane Detection

The existing lane line detection methods have achieved excellent performance in complex real world scenes, but many methods have the problem of real-time efficiency, which is crucial for the automatic driving of vehicles. LaneATT is a model based on anchors, and it is a real-time attention-guided lane detection model. At present, the model's F1 Score ranks first in Culane dataset. In this project, we train LaneATT model on Tusimple dataset using our Lab server, and do some test with real world video to see the performance of the model.

1. Model Structure

   As shown in FIGURE 17, the input frame gets the feature layer F after dimensionality reduction of the Backbone network (Backbone has three different size which donated as ResNet-18, ResNet-34 and ResNet-122, respectively). Then, Use anchors to pool the feature layer and get the local feature a_i^{loc} of lanes. Because there are 2784 anchors in total, it can basically cover the entire image. However, in the case of lane hidden, global features a_i^{glob} are also needed for better prediction. Therefore, send the local feature a_i^{loc} to attention mechanism to get a_i^{glob}, and then the model concatenates the local feature and global feature as a feature set. Finally, using this feature set, two layers, one for classification and another for regression, make the final prediction.

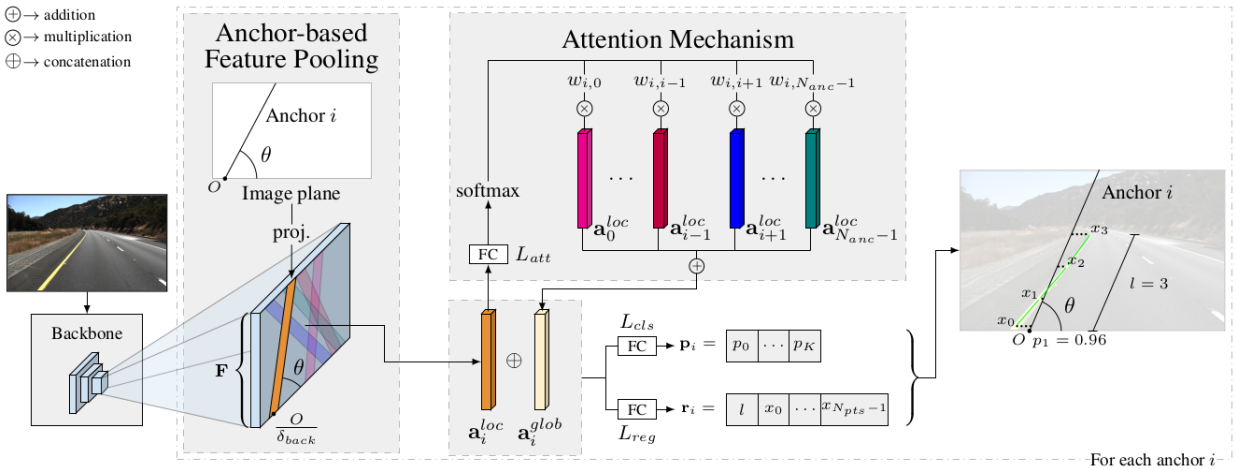


FIGURE 17. Network Structure of LaneATT

2. Innovations of the model

- Real-time

  It can reach up to 250 FPS on CULane dataset.

- Anchor based Lane Detection Attention Mechanism

  It used anchors to help find the way and introduced attention mechanism to avoid local maximum solution. Particularly, in anchors part, it generates 72 start points with 6 angles on left and right edges, and 128 start points with 15 angles on bottom edge, that is in total 2784 ($2 \times 72 \times 6 + 128 \times 15$) anchors. In FIGURE 18, we can see the angles on each edge and we have reason to believe that the super-parameters (angle) here is the most common angle obtained by the author after observing many pictures.

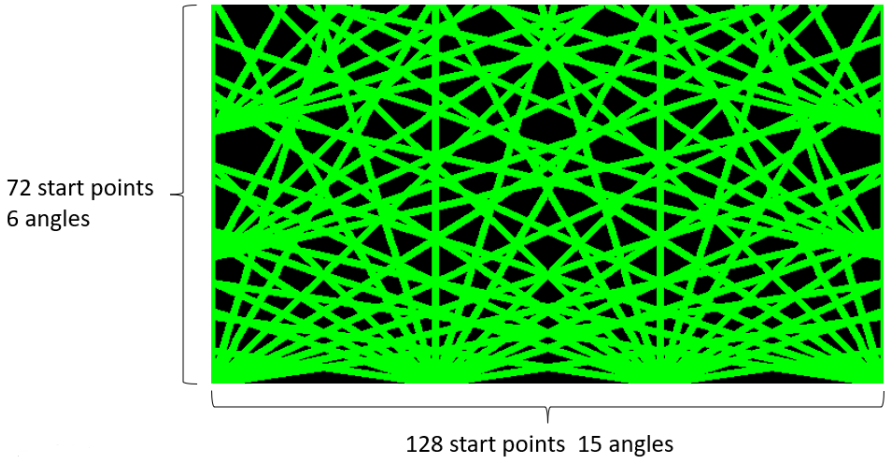


FIGURE 18. Visualize anchors: 4 start points on each edge.

3. Hardware and Software

   - CPU: Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz
   - GPU: NVIDIA GeForce RTX 2080 Super
   - Python: 3.8.15
   - Scikit-learn: 0.23.2
   - pytorch: 1.6.0
   - numpy: 1.23.4
   - CUDA: 11.6

4. Evaluation

   - Introduction to the Dataset

     The dataset is distributed into Training set and Testing set. In Training set has 3626 video clips and for each clip, there are 20 frames with 1 noted frame. As for Testing set, there are 2944 video clips. The dataset structure is shown in FIGURE 19.

   - Evaluation See FIGURE 20.

5. Drawbacks

   - Non-maximum Suppression

     According to the code, LaneATT uses nms to suppression multi-detection, but it may cause some problems when meet branch roads (FIGURE 21.(a)) whie calculating nms using intersection of union (IoU), which would cause the lost of branch road and the increase of FNR.

```
dataset
  |
  |----clips/                  # video clips
  |------|
  |------|----some_clip/       # Sequential images for the clip, 20 frames
  |------|----...
  |
  |----tasks.json              # Label data in training set, and a submission template for testing set.
```

FIGURE 19. Dataset structure of Tusimple

TuSimple

Backbone	Accuracy (%)	FDR (%)	FNR (%)	F1 (%)	FPS
ResNet-18	95.57	3.56	3.01	96.71	250
ResNet-34	95.63	3.53	2.92	96.77	171
ResNet-122	96.10	4.64	2.17	96.06	26

FIGURE 20. Accuracy and Confusion matrix on Tusimple

- Uniform sampling

  As shown in FIGURE 21.(b), when there is a large angle turn at the far end of the image, the lane fitting performs poor. That may because of the uniform sampling in the model. Since the perspective principle, the far end of the image has less pixels, which means it has less information to learn. As a result, we can improve it, that is, non-uniform sampling.

2.6 Car Distance Estimation

In this part, we use the the imaging principle of the camera and the triangle similarity theorem to estimate the distance between the front vehicles and own vehicle.

We mainly use the YOLOv5 frame to detect cars and trucks, and the model we use is yolov5s. In order to reduce the number of frames in the image (so that the whole images won't be too messy), we only keep the boxes with confidence

(a) Problem in Brand Road (b) Problem in Far End

FIGURE 21. Drawbacks of the Model

greater than 0.1.

Then, to estimate the distance between vehicles and show the results in the output frames, we only need to modify the file detect.py in YOLOV5, since we only need the information of the height of detected boxes.

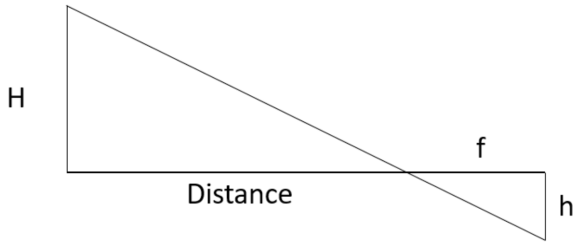


FIGURE 22. Imaging Principle of the Camera

As shown in Figure 22, given an average height of true car/truck (H) and the focal length of camera (f), we can calculate the distance using the height of boxes (h). We choose to use the height of boxes, rather than the width of boxes for our estimation, because the left and right sides of the car will be frequently blocked, which means the width of the boxes are not credible.

However, the output distance is an estimated value. Since there are no ground truth, we have no idea about whether this estimated distance is trustworthy. But with many experiments, we get a trusted focal length of lens and we use it for our final output. And the Figure 23 is one frame of our final video, the three quantities above the box represent the type of car, detection confidence and estimated distance respectively.

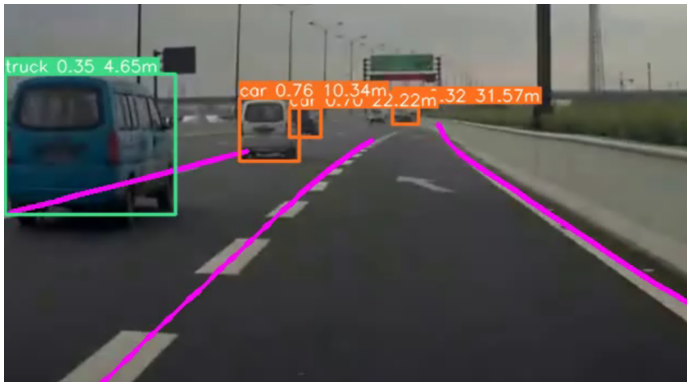


FIGURE 23. One Frame of Final Results

2.7 Staffing Work

The group members come from four different majors, and considering that each has different specialties, our group division of labor is shown in Table I.

TABLE I. Staffing Plan

Member	Major	Work
Yaqi Wang	Statistics	Code reproduction and Report Writing
Jiajian Yang	Computer Science	Lane Detection and Report Writing
Duolei Wang	mathematics	Code reproduction, Model Testing and Report Writing
Guang Chen	Electronic Science	Code reproduction, Model Optimization and Report Writing

REFERENCES

1. https://github.com/ultralytics/yolov5
2. T. -Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan and S. Belongie, "Feature Pyramid Networks for Object Detection," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 936-944, doi: 10.1109/CVPR.2017.106.
3. https://github.com/maheravi/YoloV5-TrafficSign
4. https://drive.google.com/file/d/1gQD1OovQDyjMlUEWl6IEn2mzgS6KNppX/view
5. https://github.com/ultralytics/yolov5
6. http://www.nlpr.ia.ac.cn/pal/trafficdata/recognition.html
7. Tabelini, L., Berriel, R., Paixao, T. M., Badue, C., De Souza, A. F., OliveiraSantos, T. (2021). Keep your eyes on the lane: Real-time attention-guided lane detection. In Proceedings of the IEEE/CVF Conference on ComputerVision and Pattern Recognition (pp. 294-302).