Drawing 3D bounding box in image in the absence of camera parameters

Chenhongyi Yang

hongyi@bu.edu

Outline

- Motivation and goal
- Introduction to my algorithm
- Experiment
- Using my algorithm in video

Motivation

- When processing images, sometime people want to draw a 3D bounding box to for each objects to reflect position, size, direction. To do that we need object's 3D information and camera calibration parameters like intrinsic matrix, rotation matrix.
- Now, researchers have been using deep learning methods to obtain 3D information of objects.
- However, even if we have 3D information of objects sometimes calibration parameters are not accessible. Can we draw the 3D bounding box without those parameters?

Specific goal

- Problem definition: Given an RGB image, draw its 3D bounding box on the image for each object in the image without any 3D information or calibration parameters.

- Difficulties: Different from 3D object detection, we do not have any camera parameter so we cannot use projections to draw the 3D bounding box on the image plain.

- Creativity: It seems that there is no other researcher has concerned about the problem.

My algorithm

- Use 2D bounding box and three bounding parameters to constrain a 3D bounding box in a 2D bounding box.

- 2D bounding box: 2D object detection. (Faster-RCNN, R-FCN, SSD…)

- Bounding parameters: Converting to a regression problem and training a neural network to predict them.

Algorithm

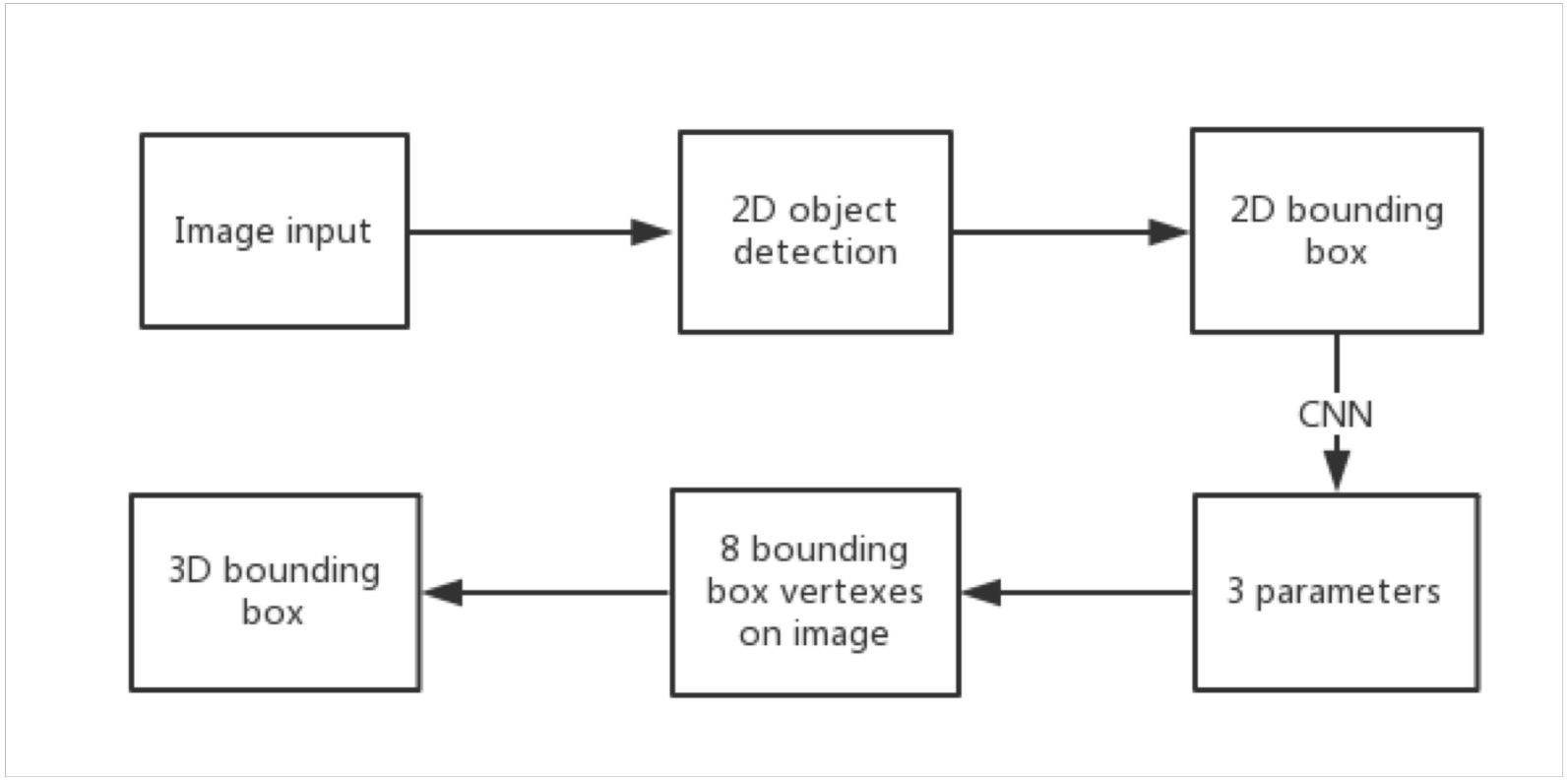
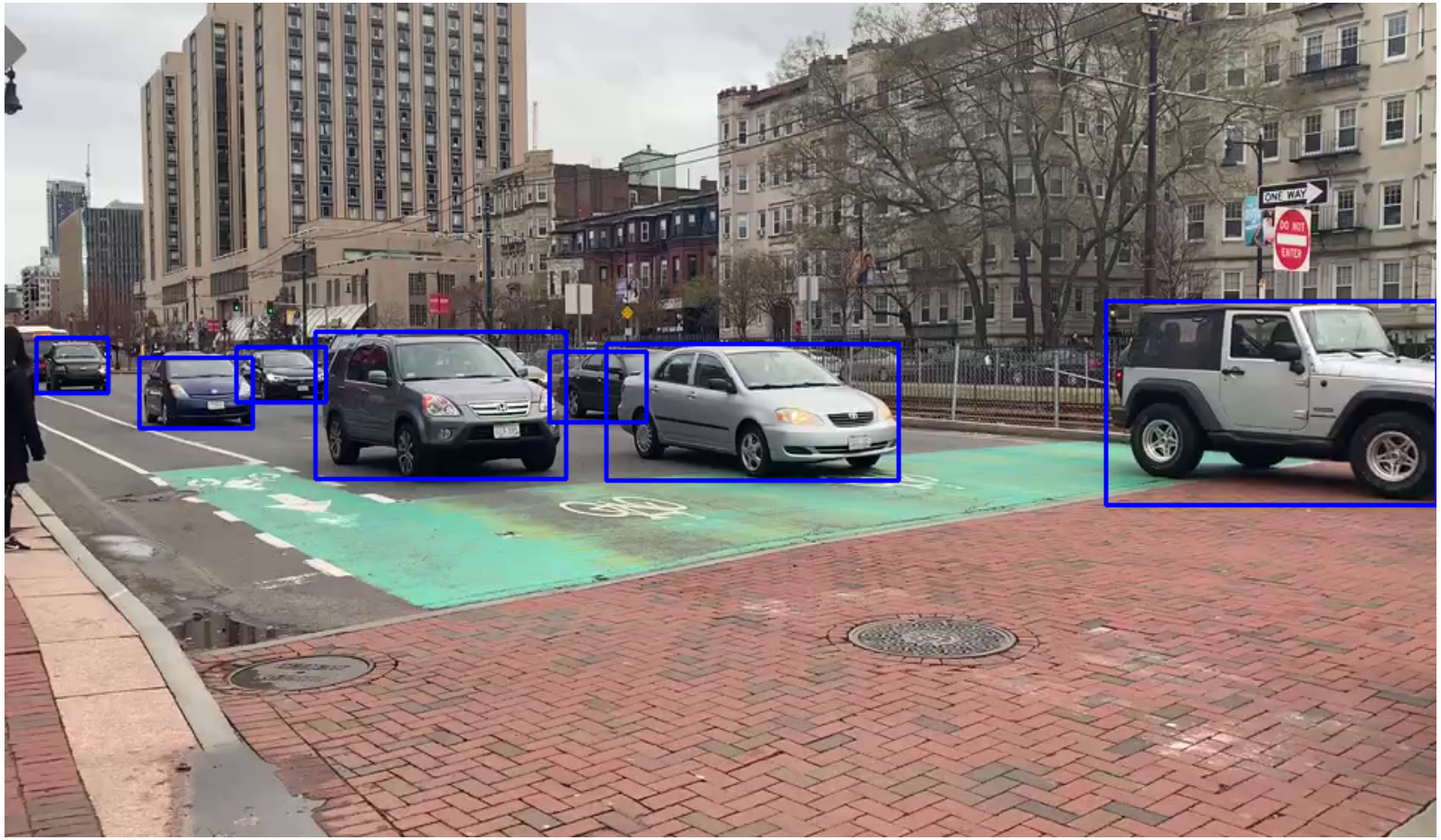
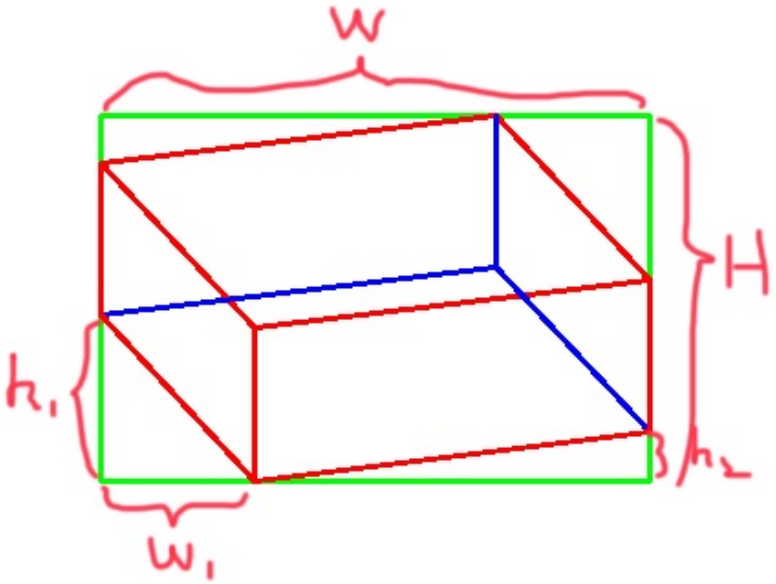


Image input
2D object detection
2D bounding box
CNN
3 parameters
8 bounding box vertexes on image
3D bounding box

2D object detection



Bounding parameters

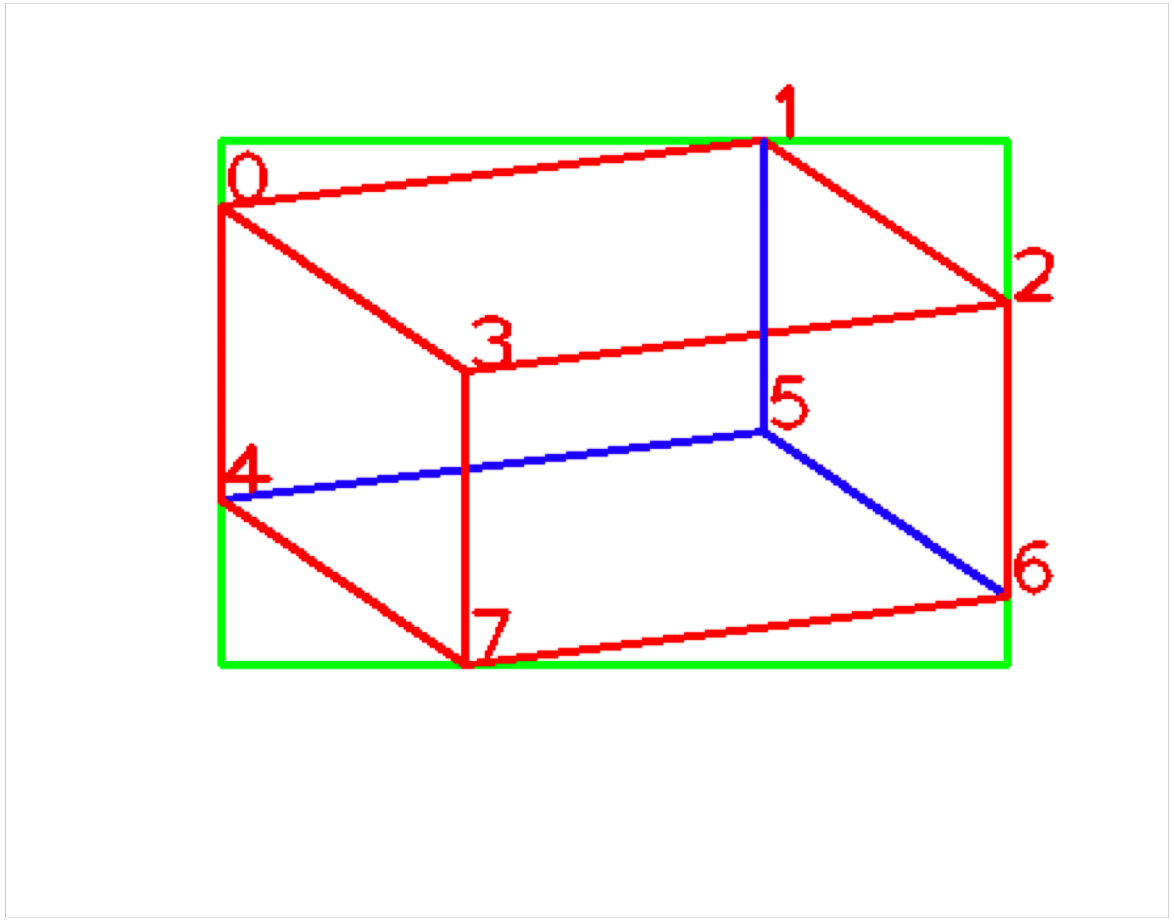


$$rs = \frac{w_1}{W}$$

$$\theta_1 = \arctan(\frac{h_1}{w_1})$$

$$\theta_2 = artan(\frac{h_2}{W - w_1})$$

Computing 3D bounding box



Vertex coordinate on 2D plain:

$$v_7 = (rs \times W, H)$$

$$v_4 = (0, H - \tan(\theta_1) \times rs \times W)$$

$$v_6 = (H, (1 - rs) \times \tan(\theta_2) \times W)$$

$$v_1 = (v_4(x) + v_6(x) - v_7(x), 0)$$

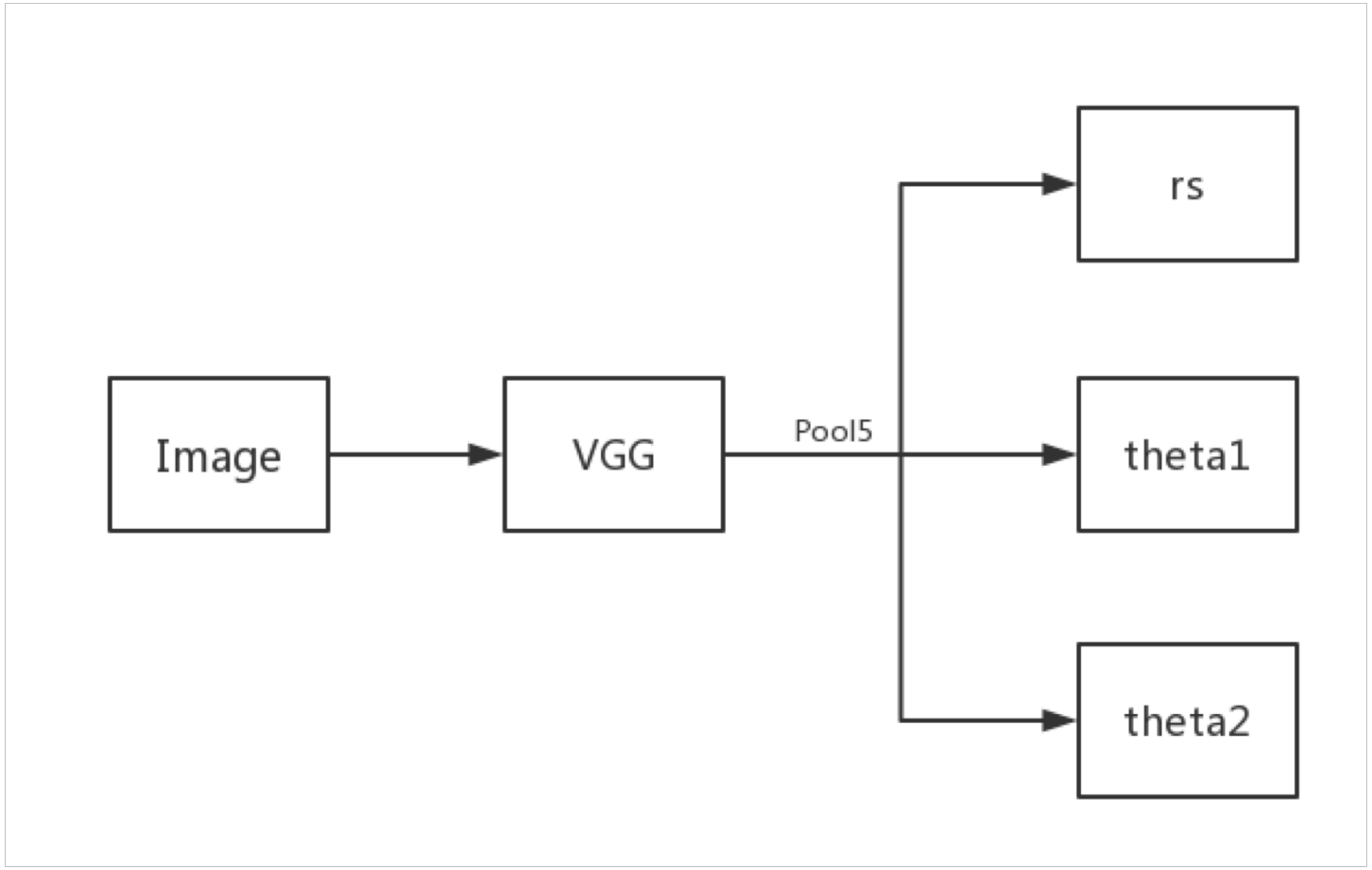
$$v_0 = v_1 + v_7 - v_6$$

$$v_3 = v_7 + v_0 - v_4$$

$$v_2 = v_6 + v_0 - v_4$$

$$v_5 = v_1 + v_0 - v_2$$

Network Architecture



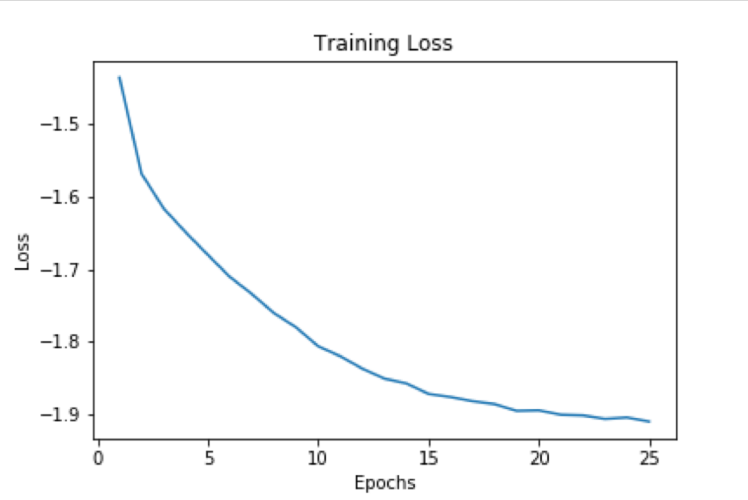
Experiment

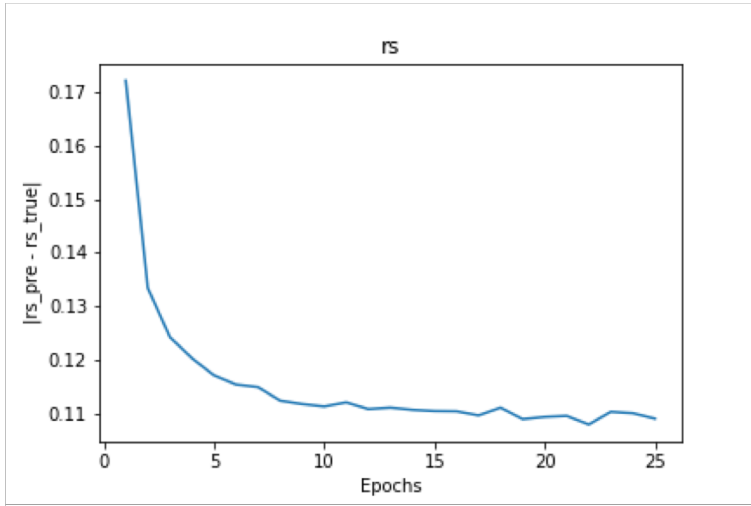
- Dataset: KITTI, 6 kinds of objects and they are Car, Truck, Van, Tram, Pedestrian, Cyclist.

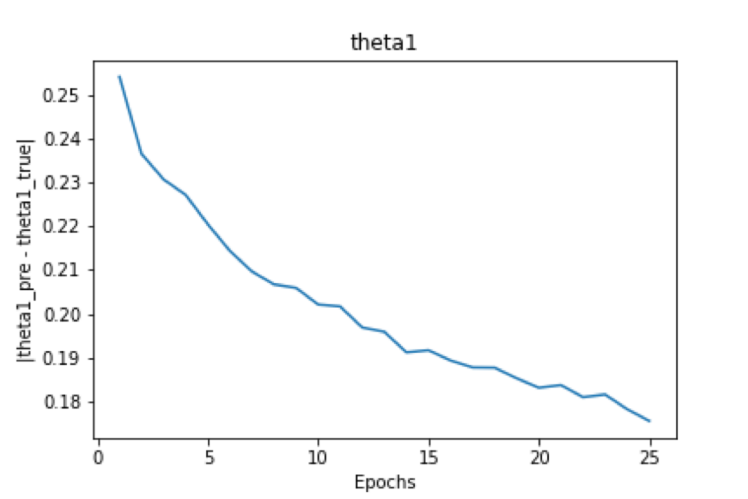
- 2D object detection system: SSD (Download from: SSD-Tensorflow)

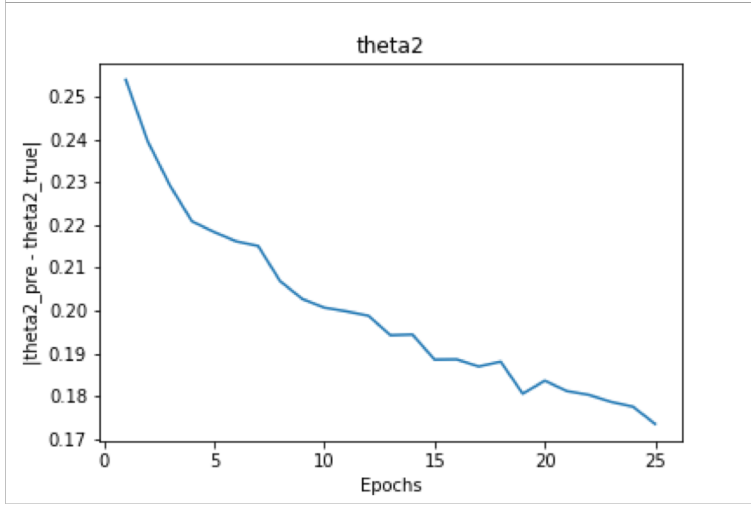
- Training: 6700 images, for each object flip it left side to right with probability 0.5.

Training

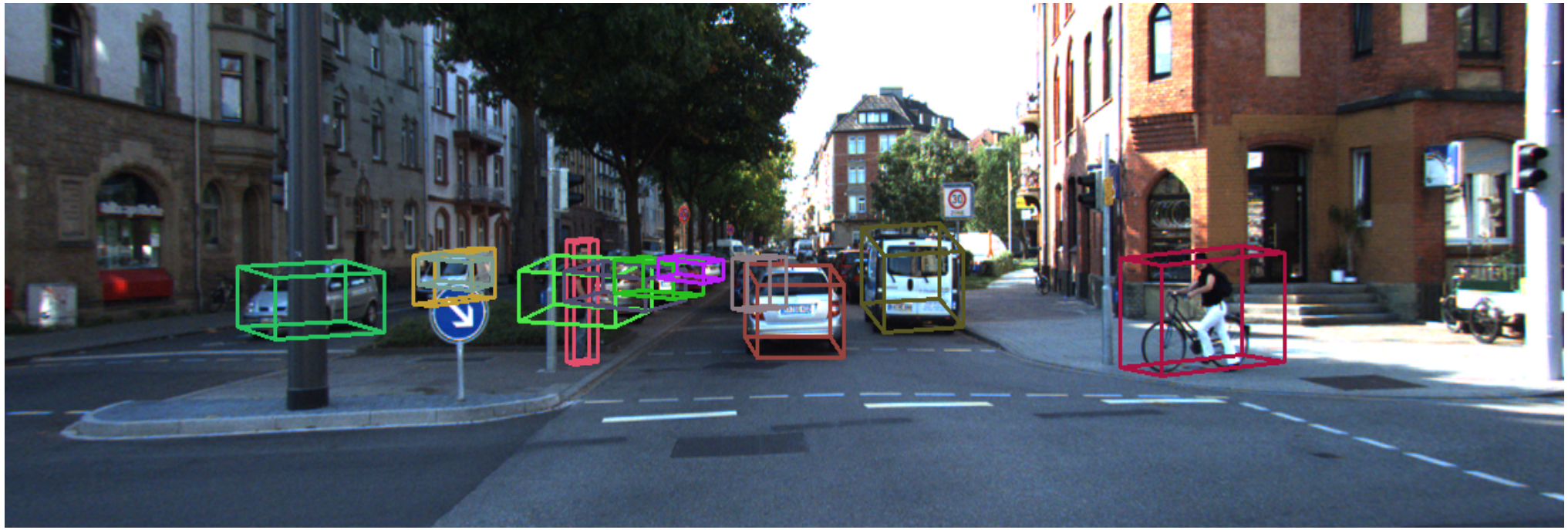

Training Loss
Loss
−1.5
−1.6
−1.7
−1.8
−1.9
0
5
10
15
20
25
Epochs


rs
|rs_pre - rs_true|
0.17
0.16
0.15
0.14
0.13
0.12
0.11
0
5
10
15
20
25
Epochs


theta1
|theta1_pre - theta1_true|
0.25
0.24
0.23
0.22
0.21
0.20
0.19
0.18
0
5
10
15
20
25
Epochs


theta2
|theta2_pre - theta2_true|
0.25
0.24
0.23
0.22
0.21
0.20
0.19
0.18
0.17
0
5
10
15
20
25
Epochs

Result



(Using ground truth 2D bounding boxes)

Evaluation

	Proportion	$\|\Delta rs\|$	$\|\triangle\theta 1\|$	$\|\triangle\theta 2\|$
Car	0.73	0.11	0.18	0.20
Truck	0.03	0.15	0.20	0.18
Van	0.08	0.12	0.16	0.18
Tram	0.01	0.22	0.23	0.31
Pedestrian	0.12	0.11	0.14	0.17
Cyclist	0.04	0.12	0.17	0.18

Test set: 783 images, 4193 objects.

The results of Truck and Tram are relatively less accurate. The reason may be that their 2D bounding box is larger than other's and their proportion is low, which make the network hard to learn.

The results of Car, Van, Tram, Pedestrian and Cyclist are good.

Using my algorithm in videos

- Problem: How to determine same objects in different frames?
- Method: Using object tracking algorithm, track not only positions and velocities, but also bounding parameters.
- Experiment: Alpha-beta filter + GNNSF(Hungarian Algorithm).

Result

- https://www.youtube.com/watch?v=CN9qLvJ5Eq8

Filmed on the BU campus.

Future work

- Map the 2D result to 3D space to do 3D object detection.
- Using my algorithm to refine the 3D object detection result.
- Find a better network architecture to achieve better performance.

Reference

- Liu W, Anguelov D, Erhan D, et al. Ssd: Single shot multibox detector[C]//European conference on computer vision. Springer, Cham, 2016: 21-37.
- Geiger A, Lenz P, Stiller C, et al. Vision meets robotics: The KITTI dataset[J]. The International Journal of Robotics Research, 2013, 32(11): 1231-1237.
- Mousavian A, Anguelov D, Flynn J, et al. 3d bounding box estimation using deep learning and geometry[C]//Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on. IEEE, 2017: 5632-5640.
- Xiang Y, Mottaghi R, Savarese S. Beyond pascal: A benchmark for 3d object detection in the wild[C]//Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on. IEEE, 2014: 75-82.

Thank you!

Code can be found at: https://github.com/ChenhongyiYang/Draw3DBox