# Final Report of Individual Study

# ResNet-based Pedestrian Classifier for On-vehicle, Real-time Pedestrian Detection System

Haolin Zhang
CITR Lab, Department of Electrical and Computer Engineering, The Ohio State University

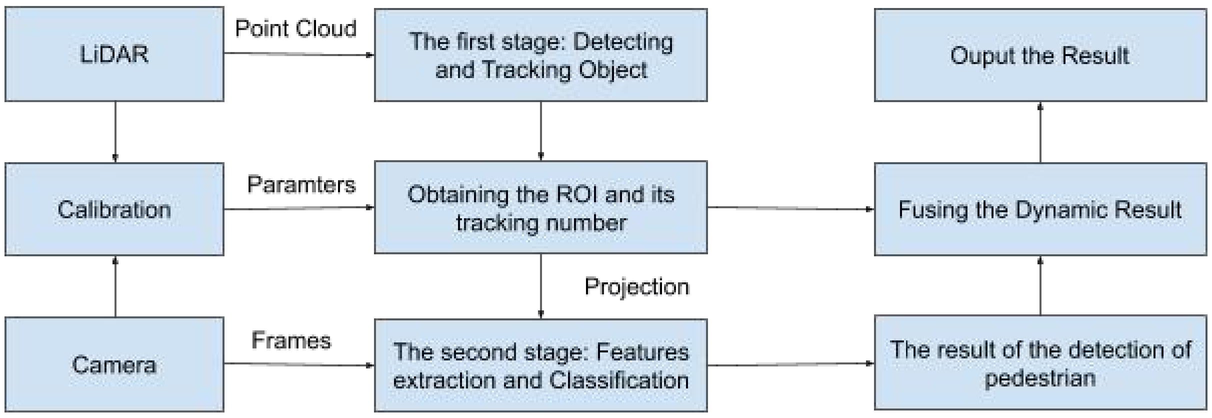
## 1 Introduction

Real-time on-vehicle pedestrian detection and prediction are crucial tasks in the application of advanced driver assistance systems (ADAS) and self-driving. The approaches and cost to accomplish these tasks vary, depending on the combination of different types of sensors. Pedestrian detection can be viewed as a sub-task or part of the general object detection, which identifies and classifies objects into various classes. Note that the focus of this work is only pedestrian detection, which may simplify the detection problem and for to which some of the object detection algorithms may not apply. However, the general ideas remain the same. That is to say, the goal of the system is to output the positions of the targets relative to the vehicle coordinate.

There are three mainstream ways for pedestrian detection: pure vision, pure point cloud, and fusion of vision and point cloud. The pure vision methods are subdivided into one-stage methods[1~6] and two-stage methods[7~11], which are mostly based on convolutional neural networks (CNNs). By one-stage methods, a whole complex network is trained to take input as the entire image. The tasks of dividing regions, predicting bounding boxes and estimating the objects are all combined in such one network. While for two-stage methods, they need two steps to complete the whole task of object detection. The detector trained for searching the proposal regions and bounding boxes is used in the first stage and then the classifier trained for estimating the objects is applied in the second stage. The second method applies multiple sensors, of which the results are somehow mixed, fused, or combined to achieve the detection task[12~18]. By such methods, the LiDAR point clouds can offer the depth information of objects, while the image could provide the color or texture information of objects. Based on different procedures, multi-information can help the system complete the task of 3D object detection more successfully. The third one uses only LiDAR point clouds, which is directly processed to obtain the pedestrian position[19~21]. Since the depth information can be collected and processed through LiDAR data directly, the procedures of object detection could be simplified.

State-of-the-art algorithms of the first method (pure vision) are mature and robust. But in essence it lacks the direct measurement of distance and depth, which restricts the detection performance in complex scenarios, for example, occluded pedestrians. Although some 3D methods that use stereo can be used to solve such problems, it will lead to more time consumption. On the contrary, LiDAR may be able to solve the problem of distance and depth faster and more accurate. Moreover, the proposal regions of objects can be predicted and located with processing LiDAR point clouds, which will be more reliable to complete the first stage of two-stage vision-based object detection methods. The third method (pure point clouds) is novel but not mature enough. Many errors will appear unpredictably in real world when only relying on LiDAR point clouds. For instance, it is difficult for systems to distinguish real people from human sculptures via point clouds since point clouds only provide the information of 3D points. However, images can offer more information such as color, bringing about better classification of objects. Therefore, the second method (the combination of point cloud and vision) is selected for our purpose, since the proposal regions generated by point clouds are more accurate and the direct measurement of distance by LiDAR provides a solid source for vehicle safety.

In our project, a real-time pedestrian detection system, which is a two-stage method based on both LiDAR point cloud and front camera vision (shown in Figure 1), was proposed. Specifically, point cloud from a 16-layer VLP LiDAR is firstly processed to generate proposal regions (candidate bounding boxes), and following that the pedestrians are identified by cropped vision from a Point Gray ethernet camera. Compared with existing approaches, our major goal is to make the system more lightweight so that the detection result can be used in real-time for pedestrian prediction, and because of two sensors combined, the results are expected to be more robust than by using only camera or LiDAR.


LiDAR
Point Cloud
The first stage: Detecting and Tracking Object
Ouput the Result
Calibration
Paramters
Obtaining the ROI and its tracking number
Fusing the Dynamic Result
Projection
Camera
Frames
The second stage: Features extraction and Classification
The result of the detection of pedestrian
Figure 1. Pedestrian detection system

In this paper, the part of work related to camera vision, which includes ROS setting, camera setting, camera calibration, LiDAR-camera calibration, dataset reorganization and classification for transportation objects (mainly for pedestrian) will be illustrated in detail, and then experimental results, further optimization and future work will be discussed.

## 2 Backgrounds

In this section, some related work and backgrounds are briefly introduced. The software resources that this project depends on, camera and LiDAR-Camera calibration methods, image classification methods and datasets for image classification are respectively introduced in Section 2.1, 2.2, 2.3, and 2.4.

### 2.1 Software Resources

For a practical engineering application, relying only on the algorithm is not enough. Instead, the selection of the platform, the computer language and the dependency libraries is also very significant, as a good choice can greatly increase the efficiency, and also provide convenience for future expansion and optimization. Thus, some excellent resources were explored: The Robot Operating System (ROS) is a set of software libraries and tools that help build robot applications [22] and it is convenient to connect different sensors and computers by publishing and subscribing messages. Pytorch [23] is a popular platform for deep learning, and a rich ecosystem of tools and libraries extends PyTorch and supports development in computer vision, NLP and more.

### 2.2 Calibration

Calibration is a very critical step in the application of sensors. Only by calibrating different sensors into the same coordinate system can users or computers understand the relative position between the targets. For camera and LiDAR, the intrinsic parameters of the camera and extrinsic parameters of the LiDAR and the camera should be found in order to restore targets to a unified world coordinate system.

Camera calibration [24][25] is based on a pinhole camera model, which is a right-handed system, with the world X and Y aligned with the image x and y (shown in Figure 2). By shifting a 8x6 checkerboard with known dimensions (shown in Figure 3) and doing linear parameter estimation under point matching, the intrinsic parameters can be obtained.

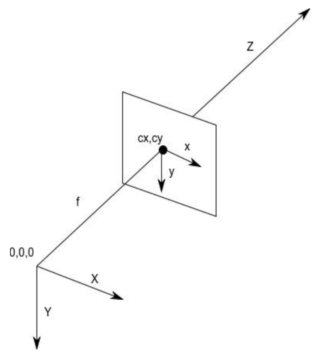


Figure 2. Pinhole camera coordinate system[24]

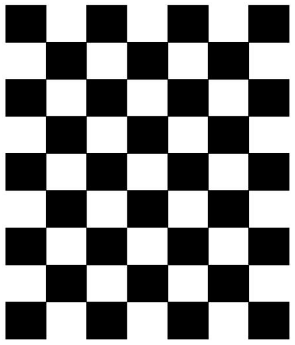


Figure 3. 8x6 checkerboard with known dimensions[24]

LiDAR-camera calibration [26][27] relies on a 3D-2D model, which is used for projecting 3D LiDAR points into 2D image points. By fixing point cloud and image at the same frame and then finding the same point (mainly choosing corners of the checkerboard for less errors) from 3D point cloud and 2D image, the extrinsic parameters can be obtained through linear estimation. Then the relationship of position between 3D LiDAR and camera can be built as shown in Figure 4.

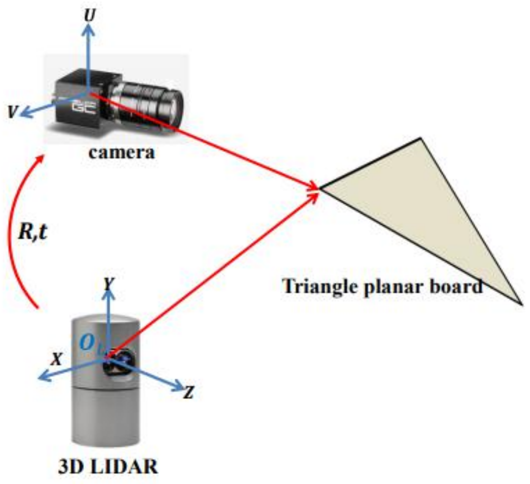


Figure 4. Calibration configuration of a camera and 3D LiDAR [26]

## 2.3 Image Classification

Image classification is the classical task and the research issue in the field of computer vision. Such task is usually accomplished by two steps.

Features extraction is the first step. The early feature extraction work is mainly based on the extraction of corner points, such as Harris corner[28] and Susan corner[29]. Since the 21st century, extracting feature descriptors such as SIFT (Scale Invariant Feature Transform)[30] and SURF (Speeded Up Robust Features)[31] have become the mainstream. In recent years, the image features based on deep learning methods [32][33] have gradually replaced the traditional features, and with the rapid development of the hardware, the time-consuming disadvantage of deep learning in features extraction has been gradually reduced. Features classification is the second step. The combination of HOG (Histograms of oriented gradients) and SVM [34] was once a popular method for image classification, and it was also applied for pedestrian recognition. However, with the development and progress of deep learning methods, the new methods by using CNN-based features to complicate classification tasks such as VGG-Net [35], GoogleNet [36] and ResNet [37] are enough to replace the traditional methods, and it is worth implementing them in the actual scene. Recently, more robust algorithms based on such improved methods such as GoogleNetV2 [38], WideResNet [39] and ResNext [40] have also been proposed, which will lead deep learning methods to be more popular and reliable in vision tasks.

### 2.4 Pedestrian Datasets

For training the classifier, it is crucial to choose desirable datasets to drive it. There are lots of publicly available datasets about pedestrian, and MIT [41], CityPerson [42], KITTI [43] and INRIA [44] are some of popular and classical choices among them online. However, such datasets contain both the bounding boxes and ground truth of classification, mainly aiming to prepare for pure vision object detection algorithms. Since the LiDAR part of work will offer the bounding boxes for the classifier, more suitable dataset should be explored for our project. MIO-TCD Dataset [45] consists of 786,702 images in the classification dataset. Those images have been selected to cover a wide range of challenges and are representative of typical visual data captured today in urban traffic scenarios. The dataset not only contains the pedestrian images, but also includes other main transportation objects. PETA Dataset [46] is also a good pedestrian dataset with large-scale pedestrian images, which could be a complement for pedestrian data. Additionally, Caltech 256 [47], Cars Dataset [48] and ImageNet [49] could also help when more training data are needed.

## 3 Methods

In this section, the work and approaches involved in the camera's part of on-vehicle, real-time pedestrian detection system (the pink part in Figure 5) are illustrated. The hardware and software setting, camera calibration, LiDAR-camera calibration, datasets reorganization and ResNet-based pedestrian classifier's build are respectively illustrated in Section 3.1, 3.2, 3.3, 3.4 and 3.5.

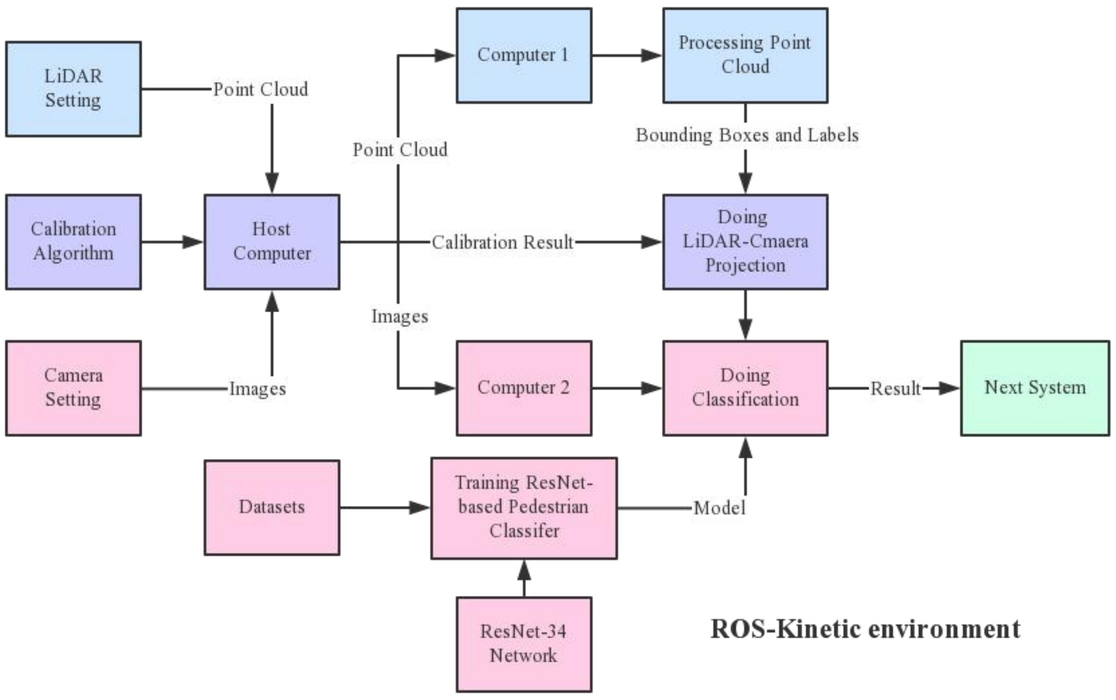


Figure 5. The pipeline of on-vehicle, real-time pedestrian detection system

## 3.1 Hardware and software setting

In camera's part, the hardware consists of  MITXPC, Netgear, Point Gray ethernet camera, two computers (one for training model and one for application) and some connections. The main function of different hardware can be seen in Table 1, and they were set in our experimental car (as shown in Figure 6).

| Hardware | Function |
|---|---|
| MITXPC | Collecting the data from sensors |
| Netgear | Networking devices |
| Point Gray ethernet camera | Steaming images |
| Desktop (with NVIDIA 2070 Super) | Training the pedestrian classifier |
| NVIDIA Jetson TX1 | Implementing algorithms the model |

Table 1. Main hardware

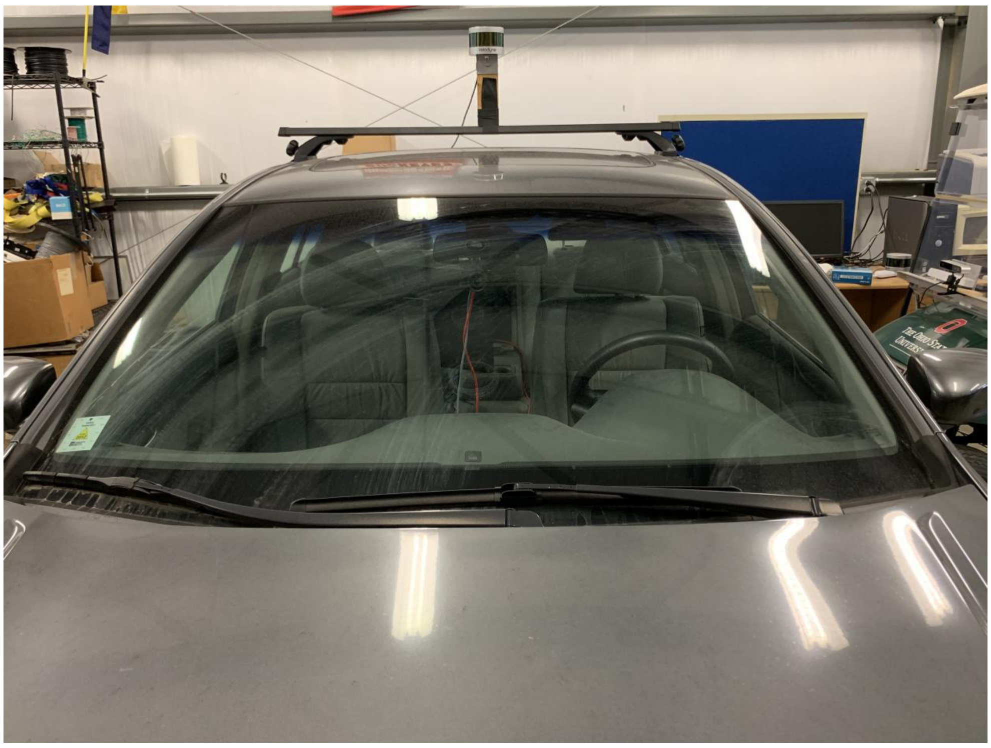


Figure 6. The experimental car

For software, the version of  Kinetic Kame (Python 2.7) of ROS was chosen for our project. Through ROS system, the LiDAR point cloud and images can be published and subscribed between different computers. Moreover, Pytorch 1.0 was chosen for the implementation of algorithms. Additionally, OpenCV, Matplotlib and some other frequently-used libraries are applied for image processing and experiments.

## 3.2 Camera Calibration

It is easy to obtain intrinsic parameters including camera matrix K=[$f_x$, 0, $c_x$; 0, $f_y$, $c_y$; 0, 0, 1] and distortion matrix D=[$k_1$, $k_2$, $p_1$, $p_2$, $k_3$] of our camera via the printed checkerboard and OpenCV tools, and such parameters can be saved as *Camera Info* in ROS system for message passing. According to distortion models (shown in equation (1) and (2) with $r^2=x^2+y^2$), the mathematical relation of distortion coordinates system ($u_d$, $v_d$) and non-distortion coordinates system (u,v) can be calculated (shown in equation (3), (4) and (5)), and thus the image can be corrected by removing the distortion from the relation.

$$\begin{cases} u' = u(1+k_1r^2+k_2r^4+k_3r^6) \\ v' = v(1+k_1r^2+k_2r^4+k_3r^6) \end{cases} \tag{1}$$

$$\begin{cases} u' = u+[2p_1v+p_2(r^2+2u^2)] \\ v' = v+[2p_2v+p_1(r^2+2u^2)] \end{cases} \tag{2}$$

$$\begin{cases} x' = (u-c_x)/f_x \\ y' = (v-c_x)/f_y \end{cases} \tag{3}$$

$$\begin{cases} x'' = x'(1+k_1r^2+k_2r^4)+2p_1x'y'+p_2(r^2+2x'^2) \\ y'' = y'(1+k_1r^2+k_2r^4)+2p_2x'y'+p_1(r^2+2y'^2) \end{cases} \tag{4}$$

$$\begin{cases} u_d = f_xx''+c_x \\ v_d = f_yy''+c_y \end{cases} \tag{5}$$

## 3.3 LiDAR-camera Calibration

The transformation from a LiDAR 3D point (x, y, z) to a 2D image point (u, v) can be represented as in equation (6). where $f_u$ and $f_v$ are the focal lengths in horizontal and vertical directions, respectively, and ($u_0$, $v_0$) is the center point of the image plane. Also, R and t are the rotation and the translation matrices. Our goal is to calculate the M transformation matrix.

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = M \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{6}$$

Instead of using auto-selective point matching methods in [26][27], manually choosing corresponding points between 3D LiDAR point cloud and 2D images at the same frame was applied for our calibration step in order to reduce errors. Sixteen corresponding points were selected for calibration at varying position and depths of the checkerboard, and then nonlinear estimation method EPnP was used to find the Rotation (3x3 matrix) and Translation (3x1 matrix) transforms between the camera and the LiDAR.

Fusing them with the intrinsic parameters (K matrix), the M matrix can be obtained, and finally the 3D LiDAR point can be projected into 2D image through equation(7), and the test result is shown as Figure 7.

$$\begin{cases} v = \dfrac{m_{11}x + m_{12}y + m_{13}z + m_{14}}{m_{31}x + m_{32}y + m_{33}z + m_{34}} \\ u = \dfrac{m_{21}x + m_{22}y + m_{23}z + m_{24}}{m_{31}x + m_{32}y + m_{33}z + m_{34}} \end{cases} \tag{7}$$

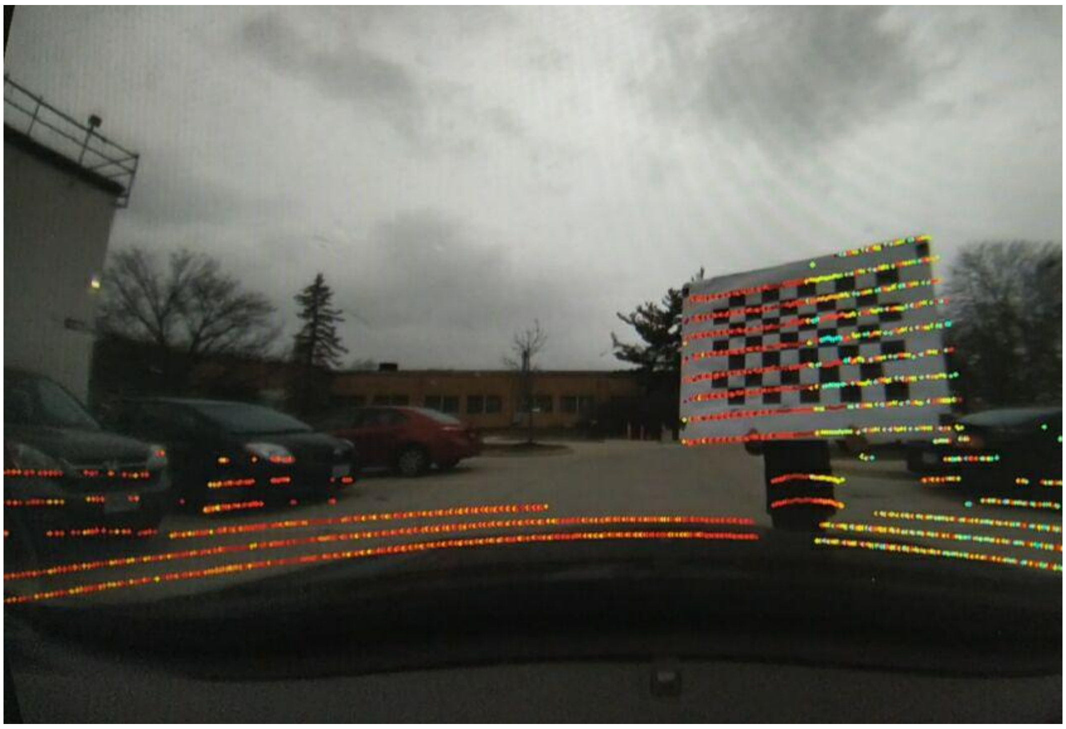


Figure 7. The result of projecting 3D LiDAR point cloud into 2D image

### 3.4 Datasets Reorganization

MIO-TCD Dataset [45] was chosen as the main dataset for our project. It contains 11 categories: Articulated truck, Bicycle, Bus, Car, Motorcycle, Non-motorized vehicle, Pedestrian, Pickup truck, Single unit truck, Work van, Background. For expending this dataset, PETA Dataset [46], Caltech 256 (mainly using the category of animal) [47] and Cars Dataset [48] were also added into it. Finally, the total dataset containing half million images was reorganized and reshaped into 10 categories including pedestrian, car, bus, non-motorized vehicle, motorcycle, bicycle, truck, work van, animal and background as shown in Table 2. The 70% of images of every category were defined as training data for classifier. The 15% of images of every category were defined as validating data for adjusting model, and the rest of images was defined as test data for carrying out experiments.

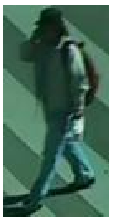
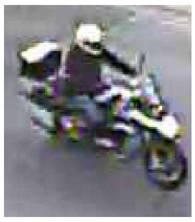
| | | | | |
|---|---|---|---|---|
| Category 0: Pedestrian | Category 1: Car | Category 2: Bus | Category 3: Non-motorized vehicle | Category 4: Motorcycle |
|  | | | |  |
| Category 5: Bicycle | Category 6: Truck | Category 7: Work van | Category 8: Animal | Category 9: Background |
| | | | | |

Table 2. The reshaped 10-categories dataset

### 3.5 ResNet-based Pedestrian Classification

The deep learning network used for our classifier was ResNet-34, and the architecture can be seen in Figure 8 or Table 3.

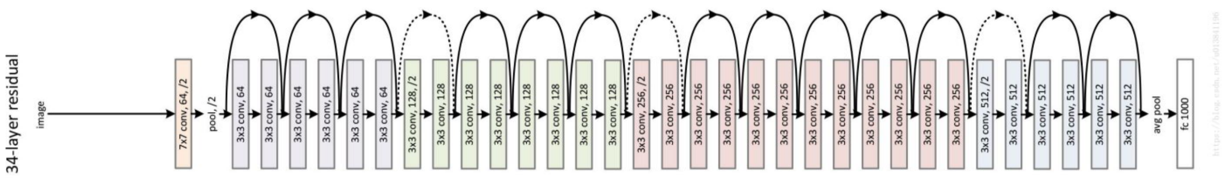


Figure 8. The architecture of ResNet-34

| **Layer name** | **Ouput size** | **ResNet-34-Layer** |
|---|---|---|
| Conv1 | 112 x 112 | 7x7,64, stride 2 |
| Conv2_x | 3 x 3 max pool, stride 2 | |
| | 56 x 56 | $\begin{bmatrix} 3\times3,64 \\ 3\times3,64 \end{bmatrix}\times 2$ |
| Conv3_x | 28 x 28 | $\begin{bmatrix} 3\times3,128 \\ 3\times3,128 \end{bmatrix}\times 2$ |
| Conv4_x | 14 x 14 | $\begin{bmatrix} 3\times3,256 \\ 3\times3,256 \end{bmatrix}\times 2$ |
| Conv5_x | 7 x 7 | $\begin{bmatrix} 3\times3,512 \\ 3\times3,512 \end{bmatrix}\times 2$ |
| | 1 x 1 | Average pool, 1000-d fully connections, softmax |

Table 3. The architecture of ResNet-34

In contrast to the common convolution neural network, ResNet has added the structure of the *Shortcut Connection*, so that the training result can still be greatly improved with the deepening of the network layers, which mainly justifies our selection of ResNet. The structure of the Shortcut Connection is shown in Figure 9. There is a branch that connects the input directly to the output. The output and the convolution output are added to get the final output. The formula is H(x) = F(x) + x, where x is the input, F(x) is the convolution branch output, H(x) is the output of the whole structure. These connections simply perform indentity mapping so it will add neither extra parameter nor computational complexity. After adding such shortcut connections, the net will be easy to be optimized than original net, and it can improve the performance of model without increasing too much complexity.

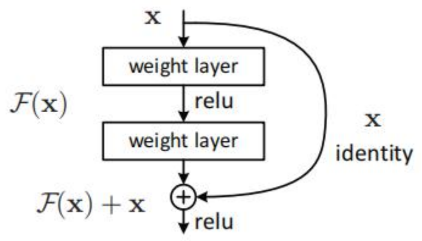

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 9. The example of shortcut connection

The methods of Stochastic Gradient Descent with Momentum and Batch Normalization were applied to optimize our model's training as engineers often use in training networks. The reorganized dataset in 3.4 was used as data input, and the model has been derived as a pedestrian classifier until the loss value decreased to about 0.1 and meanwhile the accuracy value increased to about 98% with no longer obvious change (as shown in Figure 10).

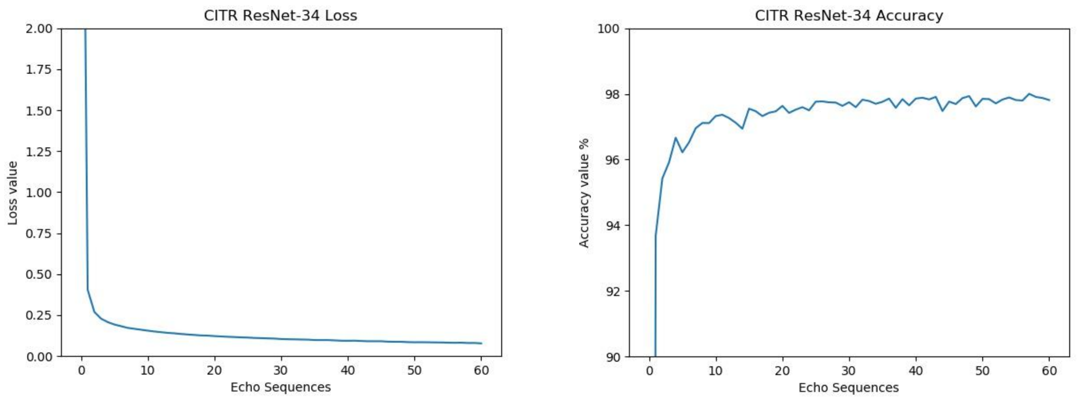


Figure 10. Curves of training loss value and accuracy value

Once 3D proposals (or 3D bounding boxes) of objects are received from LiDAR's part, the 3D proposals should be projected via 3D-2D transformation matrix into 2D images, and then the vision algorithms would process every proposal in images and the pedestrian classifier would finish the classification task, assigning a class label of every object.

## 4 Results

For evaluating our classifier's performance, experimental images from our dataset were used as test data and the confusion matrix was drawn as Figure 11 shows. A confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known. It allows the visualization of the performance of an algorithm. From the result (the label number is parallel to which in Table 2) it can be seen that label 0(Pedestrian), label 1(Car), label 2(Bus) and label 9(Background) could be well recognized since the classification accuracy of these label was close to 100%. Although the recognition accuracy of the label 4(Motorcycle), label 5(Bicycle), label 6(Truck) and label 7(Work van) was not particularly high, they did not bring false detection to label 0(pedestrian), which is the main target in our project. The drawbacks of our classifier are also obvious that the label 3(Non-motorized vehicle) cannot be precisely recognized under the 52% accuracy, and the recognition of label 8(Animal) and the recognition of label 0(Pedestrian) were sometimes confounded with each other in a small probability, which will bring potential errors to our system.

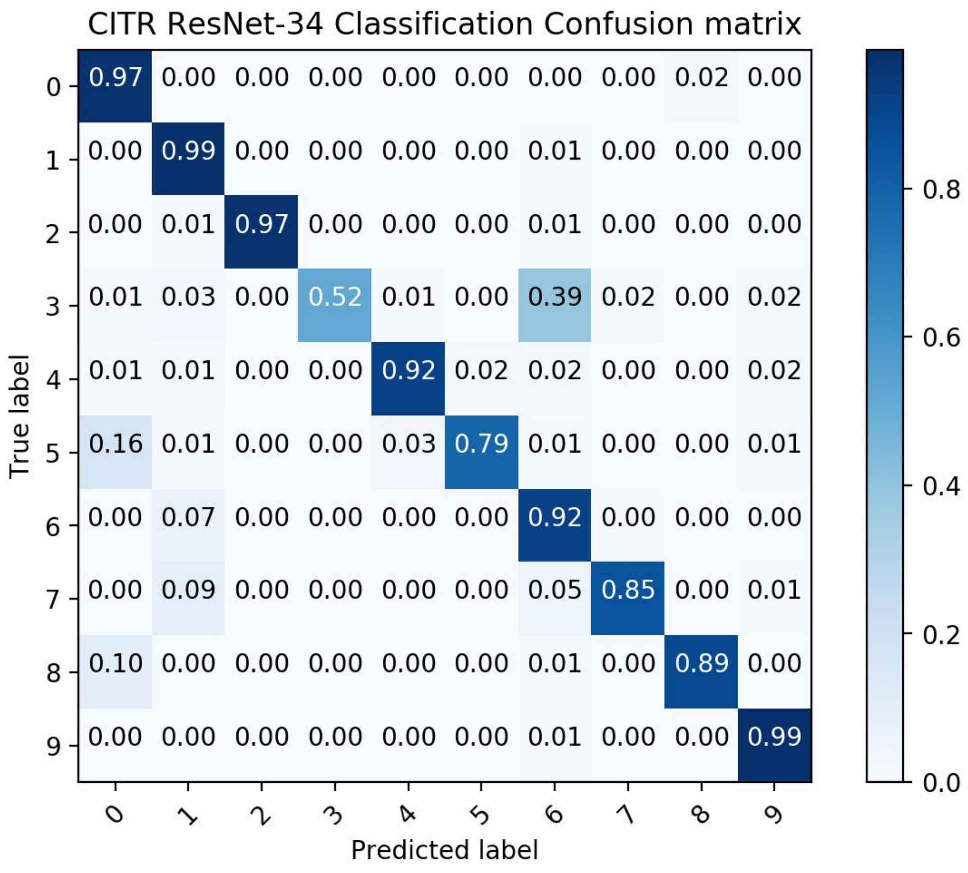
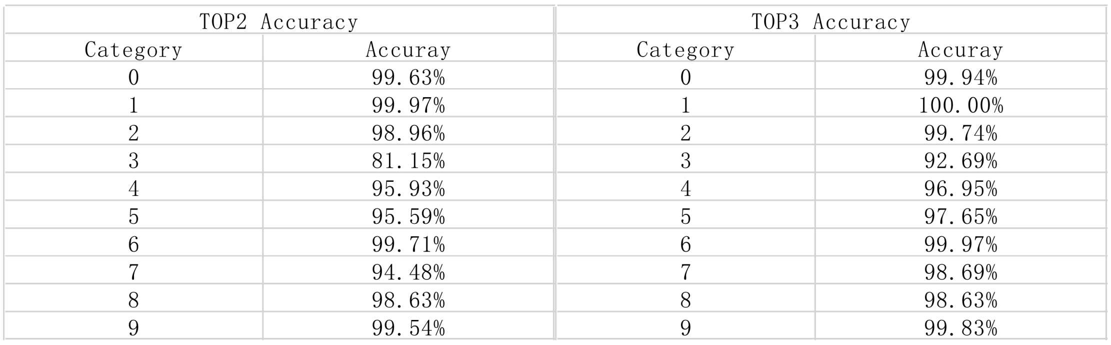


Figure 11. Confusion matrix of the experimental result

Afterwards, the Top-2 accuracy result and Top-3 accuracy result were computed as Figure 12 shows. Top-2 accuracy gave 2 highest probability answers outputted from the classifier that must match the expected answer, and Top-3 accuracy gave 3 highest probability answers outputted from the classifier that must match the expected answer. From the result it can be seen that most of the classification can be detected in the Top-2 or Top-3 with a high probability. In particular, in the first three results, the accuracy of recognizing pedestrian is as high as 99.94%.



| TOP2 Accuracy | | TOP3 Accuracy | |
|---|---|---|---|
| Category | Accuray | Category | Accuray |
| 0 | 99.63% | 0 | 99.94% |
| 1 | 99.97% | 1 | 100.00% |
| 2 | 98.96% | 2 | 99.74% |
| 3 | 81.15% | 3 | 92.69% |
| 4 | 95.93% | 4 | 96.95% |
| 5 | 95.59% | 5 | 97.65% |
| 6 | 99.71% | 6 | 99.97% |
| 7 | 94.48% | 7 | 98.69% |
| 8 | 98.63% | 8 | 98.63% |
| 9 | 99.54% | 9 | 99.83% |

Figure 12. Top-2 accuracy and Top-3 accuracy of the experimental result

Since the combination of LiDAR and camera is not well done yet, the supposed bounding boxes were manually given and sent into our classier, and there are two examples of using our classifier for pedestrian detection as shown in Figure 13 and Figure 14. TN indicates the tracking lable. PN represents the predicted label, and it is worth noting that in the second example, there was a mistake made by our classifier that the rightmost woman was detected as an animal, which demonstrates the potential error mentioned before in the analysis of the confusion matrix.

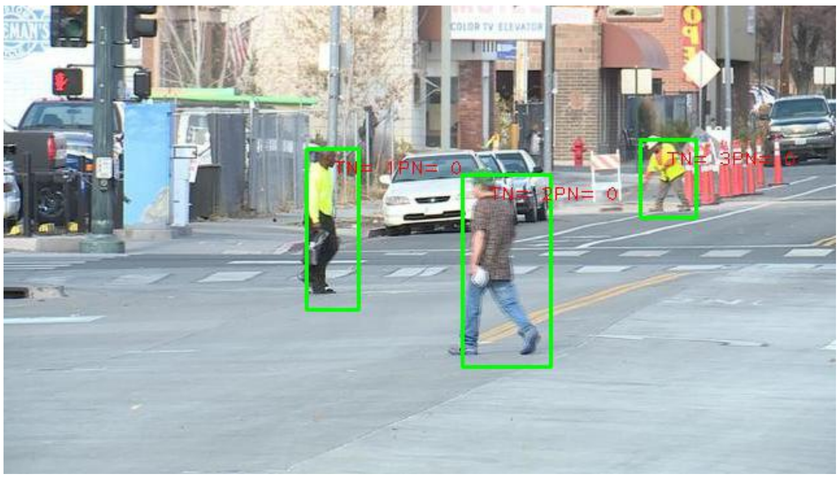


Figure 13. Example 1 of recognizing pedestrian

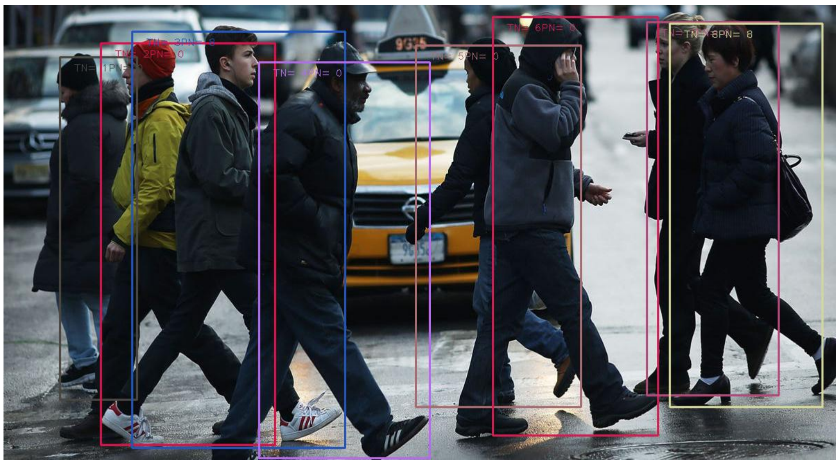


Figure 14. Example 2 of recognizing pedestrian

## 5 Discussion

There are some issues involved in our system and its results that need to be discussed so that the system could be further improved in the future work.

The first issue is about LiDAR-Camera calibration. Since the accuracy of the calibration determines the accuracy of following operations, it is important to measure the result of 3D-2D projection and attempt to improve the accuracy of projection if possible.

The second one is the recognition error made by the classifier mentioned in Section 4. Especially for pedestrian recognition, the problem of confusing the pedestrian with the animal should be figured out, and novel approach should be designed and more work would be done to better solve it.

The last one is that more potential problems may occur in real situations such as weather changes and pedestrians occlusion. In the case of rainy or snowy days, raindrops or snowflakes will not only bring a lot of noise to LiDAR's detection, but also bring uncertainty to image processing. Moreover, pedestrians occlusion will bring challenges to track and recognize pedestrian. These problems are some of the current headache problems for related research, which will also be the focuses of our project in future work.

## 6 Conclusion

In this paper, a ResNet-based pedestrian classifier and the approach of building it were introduced. For combing it with LiDAR and implementing it into on-vehicle, real-time pedestrian detection system, more work including hardware and software settings, camera calibration, LiDAR-camera calibration and datasets reorganization were accomplished first. In our experiment, the classifier has more than 97% pedestrian recognition accuracy, which seems to be a good baseline not only for testing system in real world but for further improvement.

## 7 References

[1] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection." 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 779-788.

[2] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger." 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2017, pp. 6517-6525.

[3] Redmon, Joseph, and Ali Farhadi. "Yolov3: An incremental improvement." arXiv preprint arXiv:1804.02767 (2018).

[4] Yang Z, Li J, Li H. "Real-time Pedestrian and Vehicle Detection for Autonomous Driving." 2018 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2018: 179-184.

[5] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y.Fu, and A. C. Berg. "Ssd: Single shot multibox detector." in European conference on computer vision. Springer, 2016, pp. 21–37.

[6] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar. "Focal loss for dense object detection." IEEE transactions on pattern analysis and machine intelligence, 2018.

[7] R. Girshick. "Fast R-CNN." 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, 2015, pp. 1440-1448.

[8] S. Ren, K. He, R. Girshick and J. Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137-1149, 1 June 2017.

[9] Li Z, Peng C, Yu G, et al. "Light-head r-cnn: In defense of two-stage object detector." arXiv preprint arXiv:1711.07264, 2017.

[10] K. He, G. Gkioxari, P. Dollar and R. Girshick. "Mask R-CNN." in IEEE Transactions on Pattern Analysis and Machine Intelligence. doi: 10.1109/TPAMI.2018.2844175

[11] T.-Y. Lin, P. Dollar, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. "Feature pyramid networks for object detection." in CVPR, vol. 1, no. 2, 2017, p. 4.

[12] Matti D, Ekenel H K, Thiran J P. "Combining lidar space clustering and convolutional neural networks for pedestrian detection." 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS). IEEE, 2017: 1-6.

[13] Qi C R, Liu W, Wu C, et al. "Frustum pointnets for 3d object detection from rgb-d data." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 918-927.

[14] Wang Z, Zhan W, Tomizuka M. "Fusing Bird's Eye View LIDAR Point Cloud and Front View Camera Image for 3D Object Detection." /2018 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2018: 1-6.

[15] Ku J, Mozifian M, Lee J, et al. "Joint 3d proposal generation and object detection from view aggregation." 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2018: 1-8.

[16] Cao P, Chen H, Zhang Y, et al. "Multi-View Frustum Pointnet for Object Detection in Autonomous Driving." 2019 IEEE International Conference on Image Processing (ICIP). IEEE, 2019: 3896-3899.

[17] Melotti G, Premebida C, Goncalves N M M S, et al. "Multimodal CNN pedestrian classification: A study on combining Lidar and camera data." 2018 21st International Conference on Intelligent Transportation Systems (ITSC). IEEE, 2018: 3138-3143.

[18] Kim T, Motro M, Lavieri P, et al. "Pedestrian Detection with Simplified Depth Prediction." 2018 21st International Conference on Intelligent Transportation Systems (ITSC). IEEE, 2018: 2712-2717.

[19] Kunisada Y, Yamashita T, Fujiyoshi H. "Pedestrian-Detection Method based on 1D-CNN during LiDAR Rotation." 2018 21st International Conference on Intelligent Transportation Systems (ITSC). IEEE, 2018: 2692-2697.

[20] Shi S, Wang X, Li H. "Pointrcnn: 3d object proposal generation and detection from point cloud." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019: 770-779.

[21] Yang Z, Sun Y, Liu S, et al. "STD: Sparse-to-Dense 3D Object Detector for Point Cloud." arXiv preprint arXiv:1907.10471, 2019.

[22] https://www.ros.org/

[23] https://pytorch.org/

[24] http://wiki.ros.org/image_pipeline/CameraInfo

[25] https://docs.opencv.org/3.4/d4/d94/tutorial_camera_calibration.html

[26] Park, Yoonsu, Seokmin Yun, Chee Won, Kyungeun Cho, Kyhyun Um, and Sungdae Sim. "Calibration between color camera and 3D LIDAR instruments with a polygonal planar board." Sensors 14, no. 3 (2014): 5333-5353.

[27] Dhall, Ankit, Kunal Chelani, Vishnu Radhakrishnan, and K. Madhava Krishna. "LiDAR-camera calibration using 3D-3D point correspondences." arXiv preprint arXiv:1705.09785 (2017).

[28] Shi J, Tomasi C. "Good Features to Track." IEEE Conference on Computer Vision & Pattern Recognition, 2002: 593-600.

[29] Smith S M, Brady J M. "SUSAN — A New Approach to Low Level Image Processing." International Journal of Computer Vision, 1997, 23(1):45-78.

[30] Lowe D G . "Distinctive Image Features from Scale-Invariant Keypoints." International Journal of Computer Vision, 2004, 60(2):91-110.

[31] H. Bay, T. Tuytelaars, and L. Van Gool. "Surf: Speeded up robust features." Computer Vision – ECCV 2006, pages 404 – 417.

[32] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." In Advances in neural information processing systems, pp. 1097-1105. 2012.

[33] Lin, Kevin, Jiwen Lu, Chu-Song Chen, and Jie Zhou. "Learning compact binary descriptors with unsupervised deep neural networks." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1183-1192. 2016.

[34] N. Dalal and B. Triggs. "Histograms of oriented gradients for human detection." 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 2005, pp. 886-893 vol. 1.

[35] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

[36] Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with

convolutions." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1-9. 2015.

[37] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770-778. 2016.

[38] Szegedy, Christian, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. "Rethinking the inception architecture for computer vision." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2818-2826. 2016.

[39] Zagoruyko, Sergey, and Nikos Komodakis. "Wide residual networks." arXiv preprint arXiv:1605.07146 (2016).

[40] Xie, Saining, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. "Aggregated residual transformations for deep neural networks." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1492-1500. 2017.

[41] Papageorgiou and T. Poggio. "A trainable system for object detection." Int. J. Comput. Vis., vol. 38, no. 1, pp. 15–33, 2000.

[42] S. Zhang, R. Benenson, and B. Schiele. (2017). "CityPersons:A diverse dataset for pedestrian detection." [Online]. Available:https://arxiv.org/abs/1702.05693

[43] A. Geiger, P. Lenz, and R. Urtasun. "Are we ready for autonomousdriving? The KITTI vision benchmark suite." inProc. Int. Conf. PatternRecognit. (CVPR), Jun. 2012, pp. 3354–3361

[44] N. Dalal and B. Triggs. "Histograms of oriented gradients for humandetection." inProc. IEEE Comput. Soc. Conf. Comput. Vis. PatternRecognit. (CVPR), vol. 1, Jun. 2005, pp. 886–893

[45] Luo, Zhiming, Frederic Branchaud-Charron, Carl Lemaire, Janusz Konrad, Shaozi Li, Akshaya Mishra, Andrew Achkar, Justin Eichel, and Pierre-Marc Jodoin. "MIO-TCD: A new benchmark dataset for vehicle classification and localization." IEEE Transactions on Image Processing 27, no. 10 (2018): 5129-5141.

[46] Deng, Yubin, Ping Luo, Chen Change Loy, and Xiaoou Tang. "Pedestrian attribute recognition at far distance." In Proceedings of the 22nd ACM international conference on Multimedia, pp. 789-792. ACM, 2014.

[47] http://www.vision.caltech.edu/Image_Datasets/Caltech256/

[48] Krause, Jonathan, Michael Stark, Jia Deng, and Li Fei-Fei. "3d object representations for fine-grained categorization." In Proceedings of the IEEE International Conference on Computer Vision Workshops, pp. 554-561. 2013.

[49] http://image-net.org/index