Advances in Computer, Signals and Systems (2023)
Clausius Scientific Press, Canada

DOI: 10.23977/acss.2023.070801
ISSN 2371-8838 Vol. 7 Num. 8

Research on Lane Line Detection Based on Around View System

Zhou Su[1,2,a,*], Li Keda[1,b], Zhu Xiaofeng[1,c]

[1]*School of Automotive Engineering, Tongji University, Shanghai, China*
[2]*Shanghai Zhongqiao Vocational and Technical University, Shanghai, China*
[a]*suzhou@tongji.edu.cn*, [b]*2133572@tongji.edu.cn*, [c]*zhuxiaofeng@tongji.edu.cn*
[*]*Corresponding author*

Keywords: Around view system, lane detection, semantic segmentation

Abstract: Traditional lane detection methods are limited by factors such as camera position and perspective, and often encounter issues such as false detection and missed detection. This article conducts research on lane detection methods from the perspective of multi camera BEV, and proposes a circular lane detection method based on convolutional neural networks (CNN). In order to solve the occlusion problem perceived from traditional forward looking perspectives, an around system was constructed using multiple cameras, and a multi classification semantic segmentation network was innovatively designed to predict obstructions, greatly reducing the false detection rate of obstructed lane lines. After verification, the algorithm proposed in this article can achieve good lane line detection results in different environments.

1. Introduction

In recent years, with the rapid development of autonomous driving technology, lane detection as an important part of advanced driving assistance system, has attracted more and more attention. The traditional lane line detection method is based on computer vision and image processing technology, using the vehicle's front view camera to extract lane line features for lane line detection, including image preprocessing, feature extraction, classification and segmentation. Among them, common methods include edge detection[1], color space[2], filtering[3], etc. These methods mainly rely on low-level features of images for processing, and their accuracy and stability are greatly influenced by environmental factors[4]. However, in complex scenes such as curves and intersections, traditional methods are prone to interference from factors such as lighting, shadows, and occlusion, resulting in inaccurate detection.

The around view system can solve the problem that due to the influence of viewing angle, the detection accuracy of traditional forward-looking cameras is not high. It utilizes fisheye cameras installed around the vehicle body to transform the original image into a BEV perspective through distortion removal and inverse perspective transformation[5]. Then, feature point matching and image stitching are performed on four BEV images to obtain the around view image[6,7,8]. However, the image of the around view system is based on the transformation of front, rear, left, and right camera images, rather than a real bird's-eye perspective, making it difficult to obtain the road surface situation

after being obstructed by obstacles. Therefore, for obstacles obstructing the area, the around view system is prone to false detection of lane markings.

2. Around view system

The around view system constructed in this article consists of four fisheye cameras installed in the front, rear, left, and right sides of the vehicle body, as shown in Figure 1. Generating the around image usually requires fisheye camera calibration, inverse perspective transformation, and image stitching operations.

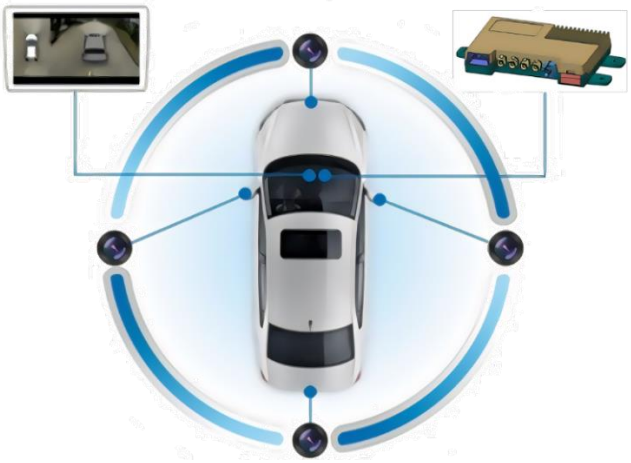

Figure 1: Composition of the Surrounding System.

2.1. Fisheye Camera Calibration

The calibration of fisheye cameras is to determine the camera's internal and external parameters, that is, to determine the correspondence between feature points in world coordinates and feature points in the image. Usually, a chessboard is used for calibration. In this article, 7 chessboards of different sizes are arranged around the vehicle body for calibration, so that each fisheye camera covers as many chessboards as possible, while the corners cover the entire image as much as possible. As shown in Figure 2:

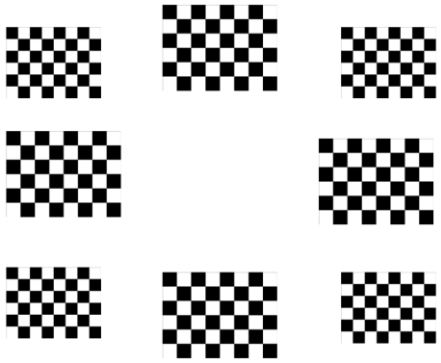

Figure 2: Checkerboard calibration board

2.2. Inverse perspective transformation

The inverse perspective transformation is the process of converting real-world points into pixel points, and its mathematical expression is as follows:

$$M^{-1}N^{-1}\begin{pmatrix}u\\v\\1\end{pmatrix}=\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\end{pmatrix}\begin{pmatrix}X\\Y\\Z\\1\end{pmatrix} \tag{1}$$

In the formula, $(X, Y, Z)^T$ represents the coordinates of the projection point in the world coordinate system, and $(u, v)^T$ represents the coordinates of its corresponding pixel coordinate system. N is the internal parameter matrix, M is the external parameter matrix, and the expression is:

$$N=\begin{pmatrix}f_x&0&c_x\\0&f_y&c_y\\0&0&1\end{pmatrix} \tag{2}$$

$$M=\begin{pmatrix}r_1&r_2&r_3&t\end{pmatrix} \tag{3}$$

In the equation, f_x and f_y are the focal length of the camera in both directions, c_x and c_y are the offset of the main point in both directions, r and t are external parameters.

2.3. Image stitching

Monography matrix is an important tool in computer vision for describing image transformations, also known as homogeneous transformation matrix. The dimension of the matrix is 3×3, as shown in equation (4), can describe the mapping relationship from one plane to another, that is, points on one plane can be mapped to corresponding points on another plane through a homography matrix. In this article, the homography matrix is used to perform BEV image stitching on the image after inverse perspective transformation. Due to the homography matrix describing transformations in homogeneous coordinates, its degree of freedom is 8.

$$H=\begin{pmatrix}h_{11}&h_{12}&h_{13}\\h_{21}&h_{22}&h_{23}\\h_{31}&h_{32}&h_{33}\end{pmatrix} \tag{4}$$

The conversion relationship between BEV subgraph and the image after inverse perspective transformation can be shown as follows:

$$\begin{pmatrix}u_{avm}\\v_{avm}\end{pmatrix}\simeq H\begin{pmatrix}u_{undis}\\v_{undis}\end{pmatrix} \tag{5}$$

In the equation, $(u_{undis}, v_{undis})^T$ is the coordinate of the image point after inverse perspective transformation, $(u_{avm}, v_{avm})^T$ is the coordinates of the pixels in the BEV subgraph, and by expanding equation (5), we can obtain:

$$h_{11}u_{undis}+h_{12}v_{undis}+h_{13}-h_{31}u_{undis}u_{avm}-h_{32}v_{undis}u_{avm}=u_{avm} \tag{6}$$

$$h_{21}u_{undis}+h_{22}v_{undis}+h_{23}-h_{31}u_{undis}u_{avm}-h_{32}v_{undis}u_{avm}=v_{avm} \tag{7}$$

Each pair of points can provide two equation constraints, and at least 4 pairs of points can be used to solve the 8 elements of the homography matrix using the Direct Linear Transform (DLT). However, to obtain more robust results, this article uses more point pairs to construct the minimum quadratic problem and use pseudo inverse methods to solve the H-matrix. Obtain the correspondence between the inverted perspective transformed image and the BEV perspective image, and then complete the stitching of the around view image.

3. Training dataset

This article uses Carla emulator to collect simulation data for training. To perform dataset collection tasks, the simulator vehicle needs to use a camera to construct around view system. Due to the lack of a wide-angle fisheye camera with a large viewing angle in the simulator, and the general RGB camera only has a horizontal viewing angle of 90°. Therefore, this article uses four RGB cameras to build the around view system. Cameras are arranged on the vehicle in a front, left, rear, and right distribution. Compared to the fisheye camera model, when using RGB cameras to concatenate BEV images, a pinhole camera model can be used. Since the external parameters of the camera can be directly obtained in the simulator, while the internal parameters can be estimated through focal length and optical center. The homography matrix used in panoramic stitching can be transformed from internal and external parameters. The acquisition frequency of the camera in this article is designed to be 30Hz, with a resolution of 800 * 800 pixels.

To obtain the semantic truth of the dataset, the vehicle is also equipped with four semantic segmentation cameras, which can provide semantic information for each pixel, such as the object type to which the pixel belongs (such as vehicles, pedestrians, roads, etc.) or the obstacle level of the pixel (such as free passage, stop signs, red lights, etc.).

This article collected a semantic segmentation dataset of 10 categories, as shown in Figure 3, which includes common semantic elements during vehicle driving, including: road surface, sidewalks, pedestrians, cars, trucks, buses, bicycles, obstacles, lane lines, and obstructions. A total of 6639 BEV semantic image data were collected. To improve the training speed of convolutional neural networks, the input images were standardized using z-score. The ratio of the total number of lane line category pixels to the total number of background category pixels in the statistical training set is approximately 1:25. Select a city map as the collection environment, and the collection scene is shown in Figure 3. Conduct a panoramic stitching of the four collected images to generate a semantic dataset from the BEV perspective.

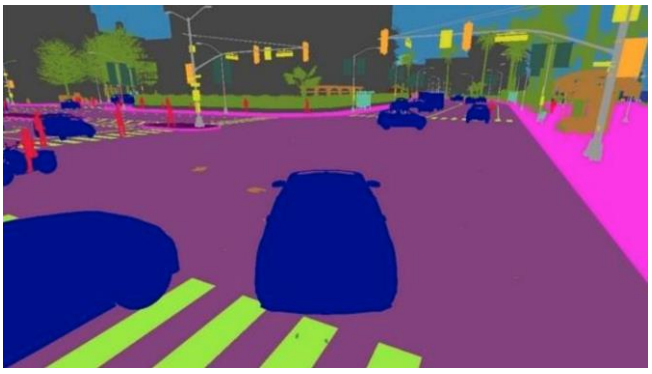

Figure 3: Dataset Example (Simulator Perspective).

To solve the problem of camera angle occlusion, this article added an extra occlusion semantic class in the data preprocessing, as shown in Figure 4. In the BEV image, for each camera's light projection area, the area obstructed by obstacles is divided into occlusion. The purpose is to enable the model to learn the features of occluded parts better, to avoid mistakenly detecting occluded parts as semantic categories of lane lines.

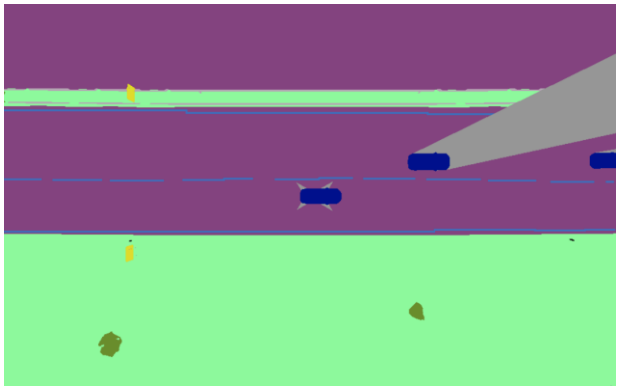

Figure 4: Example of BEV perspective obstructing object semantics.

4. Network structure

4.1. Network Overview

The semantic segmentation network is designed based on the Deeplab series of networks, which is a convolutional neural network-based image semantic segmentation method. By combining fully convolutional neural networks (FCN) with hollow convolutions (ASPP), it provides an efficient and accurate image segmentation solution.

4.2. Encoder

The encoder adopts the MobileNetV2 network, as shown in Figure 5. The network adopts Deep Separated Convolutions[9] to reduce computational complexity and parameter count, and uses residual connections to enhance feature expression capabilities.

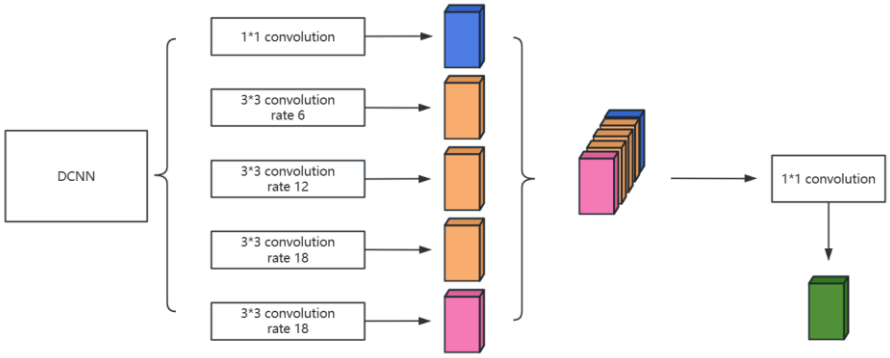

Figure 5: Encoder

The front part of this network consists of a series of deep separable convolutional layers and batch normalization layers for extracting low-level features of images. Several bottleneck blocks have been added to the back of the network to further extract high-level semantic information from the image. These bottleneck blocks include 1x1 convolutional layers, 3x3 deep separable convolutional layers, and batch normalization layers, as well as residual connections and linear activation functions.

Finally, this network uses Global Average Pooling and a 1x1 convolutional layer to compress the feature map into a vector as input to the classifier. This design can effectively reduce the number of parameters in the model and enhance its perception of global features, thereby improving the performance and generalization ability of the model.

4.3. Decoder

The decoder structure is a combination of an upsampling module and a deconvolution module. Specifically, the upsampling module uses Bilinear Interpolation[10] to upsample the feature map to twice the size, and then uses a 1x1 convolution layer to reduce the number of channels. The function of this module is to increase the resolution of feature maps and extract richer semantic information.

Then, the deconvolution module uses deconvolution and concatenation operations to sample the feature map to a size of 4 times and fuse the feature information from different levels of the encoder. Specifically, the deconvolution operation can double the number of channels in the feature map and double the size of the feature map to obtain richer semantic information. The splicing operation fuses the feature maps of different levels of the encoder to improve the semantic segmentation ability of the model.

Finally, the decoder uses a 1x1 convolutional layer and a softmax activation function to output pixel level category probability distribution for semantic segmentation.

4.4. Loss function

Tversky Loss[11] is a loss function used to train deep learning models for image segmentation tasks. Its proposal is mainly aimed at addressing the shortcomings of the cross-entropy loss function in dealing with imbalanced class problems. In image segmentation tasks, due to significant differences in the number of pixels in different categories, category imbalance is often encountered. The cross-entropy loss function often leads the model to lean towards learning a larger number of categories, while ignoring a smaller number of categories. Tversky Loss introduces two parameters (α and β, $\alpha+\beta=1$). When calculating the loss function, different categories of pixels are weighted to better handle imbalanced category issues. Specifically, its calculation formula is shown in equation (8):

$$L_{Tversky} = 1 - \frac{\sum_{i=1}^{N} p_{0i} g_{0i}}{\sum_{i=1}^{N} p_{0i} g_{0i} + \alpha \cdot \sum_{i=1}^{N} p_{0i} g_{1i} + \beta \cdot \sum_{i=1}^{N} p_{1i} g_{0i}} \tag{8}$$

p_{0i} and p_{1i} are the probability of predicting positive and negative samples for pixels in the equation, g_{0i} and g_{1i} are the probability that the true values of pixels are positive or negative samples. Use partial data in pre training to train hyperparameters α and β with different values. This article has tested the network segmentation capability, and the test results are shown in Table 1.

Table 1: Test results of different hyperparameters

hyperparameters	Presion	Recall	IoU	F-score
α=0.9,β=0.1	0.626	0.864	0.585	0.682
α=0.8,β=0.2	0.846	0.544	0.445	0.464
α=0.7,β=0.3	0.824	0.604	0.570	0.528
α=0.6,β=0.4	0.786	0.674	0.542	0.546
α=0.5,β=0.5	0.756	0.744	0.649	0.606
α=0.4,β=0.6	0.726	0.764	0.623	0.631
α=0.3,β=0.7	0.646	0.824	0.605	0.650
α=0.2,β=0.8	0.626	0.844	0.585	0.661
α=0.1,β=0.9	0.586	0.904	0.545	0.697

As shown in Table 1, when the α value is large, the segmentation accuracy of the model can reach 0.846, but the recall rate is only 0.544. At this time, the model can only segment some lane lines; When the value of β is large, the segmentation recall of the model can reach 0.904, but the accuracy is only 0.586. At this time, the model segmented most of the lane lines, but also classified many

background pixels as lane line pixels. Considering the issue of lane line segmentation comprehensively, it is necessary to be able to segment most of the lane lines, which has a high recall rate. At the same time, too many background points cannot be judged as lane lines, which means the accuracy is not low. When β =0.8, the model can simultaneously balance accuracy and recall, thus achieving relatively better segmentation results.

5. Network training

5.1. Software and hardware platforms and training processes

Table 2 shows the software platforms used in the training and testing process of the lane line semantic segmentation algorithm network model. Considering the convenience of Linux system development and related dependency libraries, this article chooses Ubuntu 16.04 operating system as the development environment, uses Python as the algorithm development language, and TensorFlow as the deep learning framework. This article also relies on the OpenCV library to implement the image processing part.

Table 2: Establishment of Experimental Platform Environment

Projects	Software platform and software library
Programming language	Python 2.8
System	Ubuntu 6.04
Deep learning framework	Tensorflow2.1
Video processing function library	Opencv4.2
Data processing function library	Numpy1.18

In model training, it is necessary to set some hyperparameters to control the behavior of the algorithm, as shown in Table 3.

Table 3: Test results of different hyperparameters

hyperparameters	value
epochs	100
learning-rate	1e-4
batch-size	5
input-size	256×512

This article uses Loss weight technology to balance the influence between different categories. This is usually achieved by assigning a weight to each category, with higher weights indicating a greater contribution of the category to the loss function. The loss weight set in this article is shown in Table 4. Among the semantic segmentation tasks for a total of 10 categories, category 1 has many pixels, while categories 3 and 7 have very few pixels. Therefore, it is possible to assign lower weights to category 1 and higher weights to categories 3 and 7, to pay more attention to the pixels of categories 3 and 7 when training the model. A higher priority has also been set for the semantic information of lane lines to ensure better training results.

Table 4: Loss Weights by Category

type	value	type	value
1 Road	0.98	6 Buses	8.41
2 Sidewalks	2.24	7 bicycles	10.91
3 Pedestrians	10.47	8 Obstacles	2.38
4 Vehicles	4.78	9 lane line	10.24
5 Trucks	7.01	10 occlusion	2.79

The loss value variation curve during the training process of the instance segmentation branch network is shown in Figure 6. The results show that Tversky Loss has a good effect on network training. During the training process, the loss value of the training set gradually decreases, and the accuracy continues to increase. It tends to stabilize after reaching the last 30 rounds of training; The change trend of the loss value is similar and converges to a lower level, without any abnormal value jumps, and it is considered that the model training has been completed.

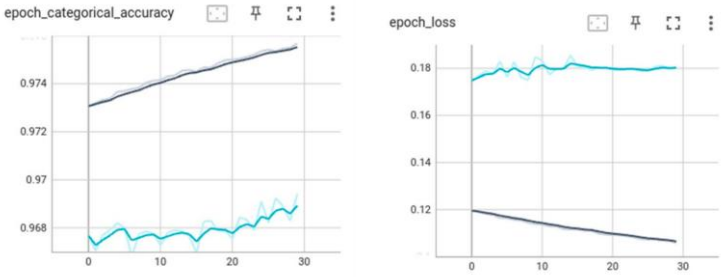

Figure 6: Changes in accuracy and loss values for the last 30 rounds.

5.2. Training results

The visualization of the heat map in Figure 7 shows the confusion matrix results, and the number of pixels in each row has been normalized row by row. Each square represents the proportion of pixels corresponding to the predicted result to the true value, and the darker the color, the higher the proportion. From the confusion matrix results, the classification results of each category are good, with misclassification mainly occurring in confusion with the background category, and the proportion of confusion between different categories is relatively small.

Figure 7: Confusion Matrix Results.

The confusion matrix can be used to conveniently calculate the intersection and union ratio of segmentation results, as shown in Table 5.

Table 5: Comparison Results of Various Types of Crossovers

type	value	type	value
road	0.970869	Buses	0.632753
sidewalk	0.920751	bicycle	0.187117
pedestrian	0.003265	Obstacles	0.861807
vehicle	0.766647	Lane line	0.963886
truck	0.613084	Occlusion	0.782592

From the value of IoU, the perception model constructed in this article has excellent semantic segmentation ability. Except for non-target semantics such as pedestrians and bicycles, the intersection to union ratio of all other categories exceeds 0.9, indicating that the network has strong perception ability for ground markings in driving areas of each category.

Figure 8 shows the labels for semantic segmentation of the validation set, with different colors used for different semantic categories. Figure 9 shows the predicted results of the visualized algorithm output from the ambient perception algorithm. The perception algorithm trained in this article exhibits good semantic segmentation performance in different environments. Although some pixels have misclassified, overall, the contour of ground elements is relatively clear, and the integrity of semantic targets is good.

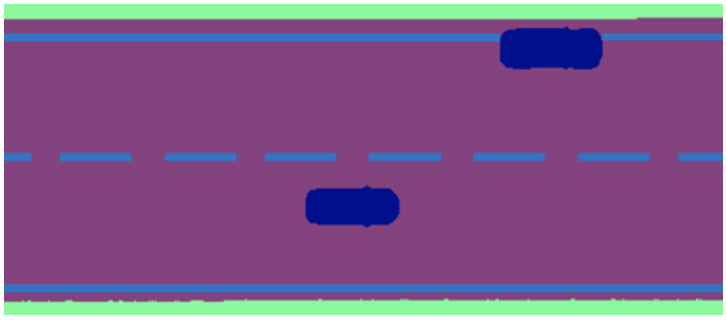

Figure 8: Example of true values for lane line datasets (unobstructed).

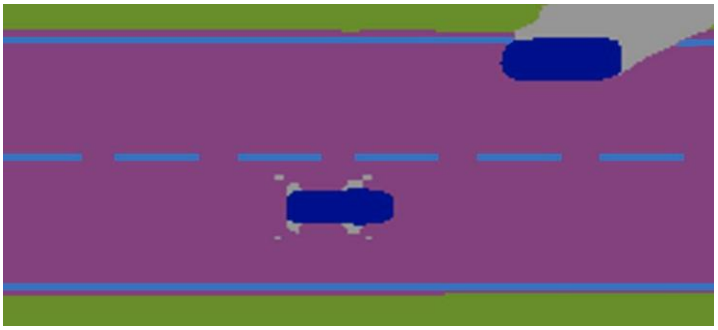

Figure 9: Example of Lane Line Prediction Results.

6. Conclusions

This article has studied the lane line semantic segmentation algorithm based on convolutional neural networks under the around view system, constructed the around view system in Carla, and collected ten classification simulation semantic segmentation datasets. Marked the lane lines and obstructed areas, and trained and tested the network; In response to the issue of imbalanced lane line pixel samples, the lane line semantic segmentation perception task from the BEV perspective was achieved by adjusting the hyperparameters of the Tversky Loss function and optimizing the CNN model using Loss weight technology. The following conclusions were drawn:

1) The lane line detection algorithm constructed in this article can effectively segment lane lines, with accuracy and recall rates of 0.626 and 0.844, respectively. It basically solved the problem of inaccurate lane line recognition by traditional forward facing cameras. However, for other non target semantic information such as pedestrians and bicycles, this detection algorithm still has shortcomings.

2) The trained network model has achieved an intersection to union ratio of 0.86 for obstacle segmentation, which can to some extent solve the problem of false detection caused by obstacle occlusion in the field of view.

References

[1] Qi L, Liu Y, Zhao Y, et al. Real-time lane detection algorithm based on edge detection [C]//2016 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER) IEEE, 2016: 67-70.

[2] Xie D, Zhang M, Xie X, et al. Lane Detection Algorithm Based on HSV Color Space [C]//2019 IEEE 4th International Conference on Image, Vision and Computing (ICIVC) IEEE, 2019: 637-641.

[3] Zare M, Gholami M, Abdi H, et al. Lane detection using the Hough transform and adaptive Gaussian filter [C]//2016 8th Conference on Information and Knowledge Technology (IKT) IEEE, 2016: 251-256.

[4] Zhang G.,&Hu W. (2019). A Review of Lane Detection Techniques. arXiv preprint arXiv: 1906.00414.

[5] J. Y. Bouguet, "Camera Calibration Toolbox for Matlab," in IEEE International Conference on Computer Vision and Pattern Recognition (CVPR) Workshop, San Francisco, California, USA, 1999.

[6] Vogel D.,&Schindler K. (2018). A Deep Network for Automatic Detection and Analysis of Road Networks in Aerial Images. IEEE Transactions on Geoscience and Remote Sensing, 56 (2), 1082-1092.

[7] Ghafoorian M., van Tulder G., de With P. H.,&Philips W. (2018). Lane detection in challenging conditions: a deep learning approach IEEE Transactions on Intelligent Transportation Systems, 20 (3), 978-990.

[8] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large Scale Image Recognition," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, pp. 1-14.

[9] Chollet F. (2017). Xception: Deep learning with depthwise separable conversions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1800-1807).

[10] Gonzalez R. C., & Woods R. E. (2008). Digital image processing (3rd ed.). Prentice Hall.

[11] Sørensen Thorkild, and Gert Stokholm Nielsen. "Generalization of Cohen's kappa to graded classifications." International conference on bioinformatics, 1997, pp. 212-220.