Spatial-Temproal Based Lane Detection Using Deep Learning

Yuhao Huang, Shitao Chen, Yu Chen, Zhiqiang Jian, and Nanning Zheng(✉)

Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, Xi'an, Shaanxi, People's Republic of China
{hyh950623, chenshitao, alanl9960212, flztiii}@stu.xjtu.edu.cn, nnzheng@mail.xjtu.edu.cn

Abstract. Lane boundary detection is a key technology for self-driving cars. In this paper, we propose a spatiotemporal, deep learning based lane boundary detection method that can accurately detect lane boundaries under complex weather conditions and traffic scenarios in real time. Our algorithm consists of three parts: (i) inverse perspective transform and lane boundary position estimation using the spatial and temporal constraints of lane boundaries, (ii) convolutional neural networks (CNN) based boundary type classification and position regression, (iii) optimization and lane fitting. Our algorithm is designed to accurately detect lane boundaries and classify line types under a variety of environment conditions in real time. We tested our proposed algorithm on three open- source datasets and also compared the results with other state-of-the-art methods. Experimental results showed that our algorithm achieved high accuracy and robustness for detecting lane boundaries in a variety of scenarios in real time. Besides, we also realized the application of our algorithm on embedded platforms and verified the algorithm's real-time performance on real self-driving cars.

Keywords: Lane detection · Inverse perspective transform
Convolutional neural network · Spatial-temporal correlation

1 Introduction

In recent years, autonomous driving has received widespread attention. Identifying traffic scenes around autonomous vehicles is a key step in autonomous driving. Besides, the identification of traffic markings in traffic scenes, especially lane boundaries, provides the necessary information for decision making in autonomous driving modules like path planning. We all know that human drivers rely mainly on vision when driving cars and the vision sensors are low-cost in the market. Therefore, the vision-based lane detection algorithm is the mainstream direction of lane detection. While there already exist plenty of vision-based lane detection algorithms, several tough challenges remain to be handled in practice. Firstly, poor lighting conditions makes it difficult to recognize lane markings at night or in other bad weather conditions like rain, snow, fog, etc. Secondly, com- plex traffic environments, including wear and tear of lane boundaries, congestion of urban roads and lane boundaries being partially

L. Iliadis et al. (Eds.): AIAI 2018, IFIP AICT 519, pp. 143–154, 2018.
https://doi.org/10.1007/978-3-319-92007-8_13

obscured, have increased the difficulty of lane detection. The last is the boundary type recognition. Compared to Advanced Driver Assistance Systems (ADAS), autonomous driving requires not only accurate lane boundary location but also precise boundary type classification.

Lane boundary detection is generally divided into three parts [6, 11]: preprocessement of the image and segmentation of the region of interest (ROI), extraction of lane boundary features, and finally lane fitting. Traditionally, lane detection has been performed by using hand-crafted features like gradient information and intensity change information [8, 12, 16]. These algorithms achieve high accuracy on highways with well-lit scenes while having a comparatively bad performance on finding lane markings when lane boundaries are obscured or the experiments are run in bad weather conditions. This is because that lane boundaries have different features in different traffic scenarios, while the hand-crafted features cannot meet the requirements of multiple scenes at the same time.

With the development of artificial intelligence, deep learning is widely used in various fields. AlexNet, VggNet [18], GoogleNet [19] and ResNet [5] achieved good results in the ImageNet Challenge from 2012 to 2015 respectively. CNN- based object detection methods like Faster R-CNN [15], YOLO [14] and SSD [9] have gained high accuracy in object detection task. Some algorithms utilized CNN as a feature extractor to identify lane boundaries and also achieved high accuracy in data sets of different conditions [8, 13, 4]. However, they are computationally complex and are difficult to be applied to driverless systems with limited computing resources.

To deal with the above challenges, we propose a real-time, robust and accurate lane detection approach. We use CNN to accurately extract the features of lane boundaries and obtain the lane boundary position in different traffic scenes, instead of the traditional lane detection algorithm which uses image processing. In order to reduce the computational complexity, we combine the constraints of spatial and temporal information of lanes to reduce the scale of CNN network structure and to ensure that our algorithm can run accurately and robustly on a self-driving platform in real time.

2 Related Work

Lane detection algorithms can be divided into two types, one includes traditional methods which use hand-crafted features like the gradient information and the color space for lane detection. Another one contains CNN-based methods that are more frequently used by researchers because of the rise of deep learning.

Since lane boundaries have color and shape informations that are different from that of the road, there are many gradient-based approaches developed to extract lane boundary features by calculating the gradient information in images and also to determine lane boundary locations. Aly et al. [1] used Hough trans- form to detect lane boundaries. In [17], hierarchical Hough transform was used by authors to detect lane boundaries with the corresponding intensity variations.

Because of CNN's powerful feature extraction capabilities, many researchers use CNN for lane boundary detection. He et al. [4] used lane boundary detection based on the front view and the top view. It first extracted lane boundary candidates using

weighted hat-like filtering in the top view and then classified lane boundary candidates according to Dual-View Convolutional Neutral Network (DVCNN). Lee et al. [9] proposed VPGNet based on multi-task learning to accomplish lane detection and ground traffic identification. In the process of lane detection, VPGNet first divided the image into 8 × 8 grids and determined if each grid belongs to a lane boundary class, and then they used sub-sampling and clustering algorithm to extract lane boundaries from the VPGNet's output. Pan et al. [14] proposed a novel network structure called SCNN, which introduced spatial information into the network structure and accurately identified lane boundaries, even in the case of lane boundary obstructions. However, the CNN network structures in these three methods are complicated and require a large amount of computing resources, making it difficult to use them in actual autonomous vehicle applications. Chen et al. [3] also used regression to do lane boundary detection. This article proposed an end-to-end method which used front view images as input and got the lane boundaries position directly from the CNN. However, it was greatly influenced by the environment.

3 Spatial and Temporal Based Lane Boundary Detection

The workflow of our algorithm is shown in Fig. 1: first, we use inverse perspective transform to get top view images and use the temporal and spatial relevance of a lane to get its estimated location. Then we cut the image into 10–20 sub-images centering on the estimated position of the lane boundary, each sub-image contains local lane boundary information. We then use CNN to classify and regress sub-images to get the exact location and category of local lane boundaries. Finally, we combine the local lane boundary location with the linear information and through the spline fitting, we can get the lane boundary information for the entire image.

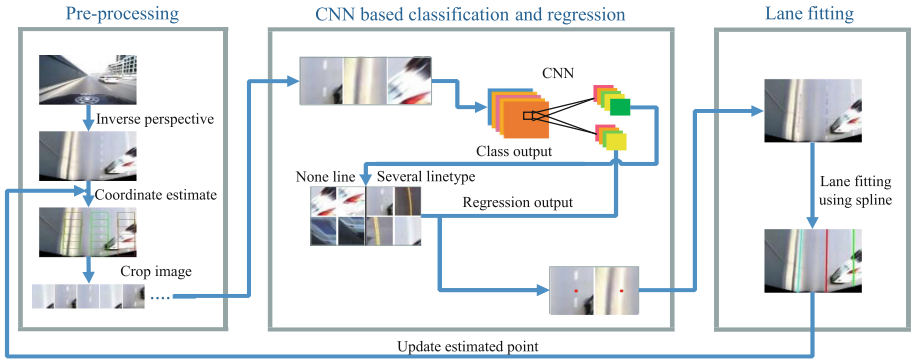


Fig. 1. Our method's workflow consists of three parts. First we perform the inverse perspective transform and estimate the rough position of lane boundaries. Next we use CNN to classify the boundary type and regress the lane boundaries position. Finally, we optimize the output of CNN and perform lane fitting.

3.1 Spatial-Temporal Based Lane Boundary Position Estimation in Top View

Inverse Perspective Transform: It is common sense that images taken by a camera have perspective effects which will destroy the original geometry of lane boundaries. Therefore, the first step in our algorithm is to convert the front view to the top view by inverse perspective mapping. Inverse perspective transform can eliminate the distortion caused by perspective effects in the picture. Through this transform step, we can get the top-down projection of the road. We select the top view as the basis for lane detection. Then we utilize the inverse perspective transform method proposed in [7]:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{z} \begin{bmatrix} f & 0 & u_c \\ 0 & f & v_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\theta_x & -sin\theta_x & y_c \\ 0 & sin\theta_x & cos\theta_x & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{1}$$

(x, y, z) is the world coordinates and (u, v) is the pixel coordinates of the image. The camera tilt is θx and the focal length is f. (uc, vc) is the camera optical center and yc is camera height. Figure 2(a) is an image of the real physical scale obtained by the transformation. For convenience, we have resized the image to 1280 × 1024. The image we used is shown in Fig. 2(b).

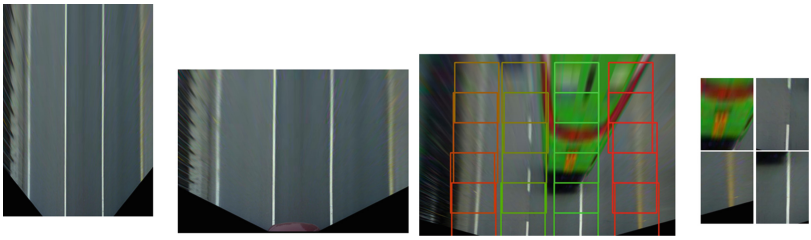


Fig. 2. (a) Is the top view image of real physical scale. (b) is the resized image. (c) shows the searching area, where the rectangles are the searching area and the estimated lane boundary positions are in the center of the rectangles. (d) illustrates some sub-images extracted from the whole image.

Coordinate Estimate: Both Lee et al. [9] and Pan et al. [13] used the single-frame lane detection methods. The benefit is that these methods don't require any prior information in the process of lane detection task, which can be completed only by inputting the image of a single frame. However, single-frame algorithms have high time and space complexity and redundancy. In application, lane boundaries have strong spatial and temporal constraints, especially in the top view. A lane boundary is a continuous ground road marking for dividing different lanes. In practice, the camera typically samples the pavement information at 30 frame per second (FPS). By

combining these two points, it is not difficult to conclude that the lane boundary position between upper and lower frame images does not change much, which means that the lane boundary in the next frame image is near the lane boundary of the previous frame. Generally, speaking, the lane width is the same for multi-lane roads, and the space constraints are obvious in the top view.

Based on the above considerations, we propose to use the temporal and spatial constraints of lane boundaries to reduce the range of searching area (shown in Fig. 2 (c)). The main difference between our detection algorithm and the algorithm in [13] is that ours is based on multi-frame image lane detection. We use the historical lane boundary information of the previous frame image as a priori knowledge to estimate the lane boundary position of the current frame. We then use CNN to accurately detect the position of the boundary based on the estimated lane boundary position with the time constraints of the lane bound- aries discussed above. The mean value of the former 10 frames' lane boundary positions is utilized as the estimated position of the current frame.

Sub-image Extraction: In our algorithm, we use control points to represent lane boundaries and a lane boundary is usually represented by five control points. Typically, we select 10–20 control points according to the boundary type classification result. These control points represent the lane information of the current lane and the lane beside the current lane. After we obtain the estimated position of the lane boundary through the above steps, both the lane boundary regression and the boundary type classification are performed in the neighbor- hood of the estimated position. So at this step we will divide the image into 10–20 sub-images centered on the estimated location and pass them to CNN (as shown in Fig. 2(d)). We use the estimated position of the lane as a priori information and cut the sub-image centered on the estimated position. After obtaining the exact location of the lane boundary through CNN, we use the exact position we get as the control point for the current frame and also as a reference for the next frame. The sub-image size needs to be determined by the camera's internal and external parameters. We consider the following rules when determining the sub-image size: first, each sub-image can only contain one lane which needs to be as large as possible. Second, sub-images should be long enough in the vertical direction to avoid lane misdetection of the white dotted boundary. However, an excess of vertical length will result in a decline in the accuracy of the curve section. In experiments, the size of the sub-image in our algorithm is 220×300.

3.2 CNN for Boundary Type Classification and Lane Boundary Regression

Network Structure: Inspired by the network structures in [2, 15], we proposed a novel convolutional neural network based on multitasking, which can accurately track the lane boundary position while classifying lane boundary types. For each task of classification and regression, the network consists of 9 layers, including 5

convolutional layers and 4 fully connected layers. Before entering CNN, we normalized the input image. The feature of our network structure is that the sub-tasks share the 5 convolutional layers and a fully connected layer, thus compressing the parameter space. Experiments show that our CNN based on multitasking can improve the performance of line boundary classification and lane regression tasks during training and eventually converges well. Our CNN is structured as follows: The first 3 convolutional layers we use the 5 × 5 kernel with 2 × 2 stride, and it followed with 2 convolutional layers which have 3 × 3 kernel and non-stride. The last convolutional layer is connected with a fully connected (FC) layer which has 4096 neurons. Then the network is divided into two parts (for the two sub-tasks of linetype classification and lane boundary regression), for each contains the same scale of 3 FC layers. The 3 FC layers have 1000, 500, 100 neurons respectively. We added a softmax layer for the sub networks which we used as the linetype classifier.

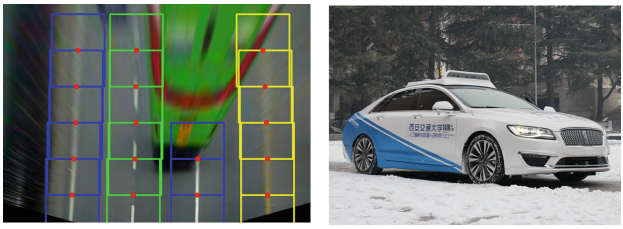
In our algorithm, we use a multitasking learning mechanism to learn and predict simultaneously the lane boundary type and the lane boundary position. Our loss function is defined as follows:

$$loss = \begin{cases} L_{cls}, boundarytype = Noneline \\ \alpha L_{cls} + (1-\alpha) L_{loc}, boundarytype \neq Noneline \end{cases} \tag{2}$$

where α is a factor to balance the classification loss and regression loss. L_{cls} *and* L_{loc} are respectively the classification loss function and regression loss function. In this paper, we use the cross-entropy loss as the classification loss function and the mean square error loss as the regression loss function.

Linetype Classification and Lane Boundary Regression: In autonomous vehicles, linetype classification is necessary since self-driving cars need to change lanes and sometimes even at a high frequency. Traditional methods are not good at boundary classification while CNN can realize it easily with its powerful classification capabilities. Based on lane categories, we classify lane boundaries into double yellow lines, white solid lines, white dashed lines, single yellow lines, Noneline, etc. The Noneline category is designed to improve the robustness of the system, as lane boundary information for lane boundaries, especially for adjacent lanes, tends to be obstructed by moving vehicles in a realistic traffic environment. In this case, the control point of the corresponding position will be set to Noneline type. We will not refer to the CNN coordinate regression value of this control point. Generally, the lane boundaries are occluded 1–2 control points, so we can still get the lane information accurately by using lane fitting.

Different from the methods which transform the regression problems into classification task, we predict the exact value of the lane boundary location directly from the CNN. There are two main benefits of using lane boundary regression: a high regression accuracy which can reach pixel level and a direct regression which requires low space and time costs with a small amount of computation. The CNN output is shown in Fig. 3(a).



(a) (b)

Fig. 3. (a) The regression and classification outputs from CNN. The red points are the regression output and the rectangles refer to the boundary type classification output, where blue, yellow and green ones represent the white solid line, yellow line and white dashed line respectively. (b) Experiment platform. (Color figure online)

3.3 Lane Fitting and Optimization

Lane Fitting: The algorithm of the previous step gives information about the control points of the current lane and its adjacent lane, including the boundary type and the exact location of the lane boundary. In this step, we fit the lane boundary through control points with the Catmull-Rom (CR) spline [20]. The CR spline is also known as Overhauster spline, which is one kind of cubic interpolation spline. It is calculated as follows:

$$P(s) = \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\tau & 0 & \tau & 0 \\ 2\tau & \tau-3 & 3-2\tau & -\tau \\ -\tau & 2-\tau & \tau-2 & \tau \end{bmatrix} \begin{bmatrix} P_{i-2} \\ P_{i-1} \\ P_i \\ P_{i+1} \end{bmatrix} \tag{3}$$

where P_i, P_{i-1}, P_{i+1} are the current point, the previous point and next point on the spline respectively. τ is the tension parameter which affects the spline's sharpness. The value of u is set in the interval (0, 1), which represents the points between P_i and P_{i-1}.

Optimization: Searching for lane boundary information throughout the whole image is unnecessary. In our algorithm, we will verify the linetype of the adjacent right and left lanes. If the left lane or the right lane cannot be changed, we will cancel the lane boundary position estimation of the adjacent lane, and in the next frame If the current lane can be changed lanes (e.g. the left lane is white dashed boundary), we will initialize the corresponding adjacent lanes based on the lane constraints. We noticed that multi-lane roads have the same lane width in general. Therefore, by calculating the deviation between the current lane's boundaries on each side, we can estimate the lane's position relative to the adjacent lane. Then in the next frame's prediction we can detect lane information for the corresponding adjacent lane. In this way the amount of computation will also be minimized.

4 Experiments

In this section, we have performed a series of contrast experiments for verifying the effectiveness of our proposed algorithm. These experiments are mainly executed on three different open-source datasets containing various traffic scenarios as well as different data:

1. Road Vehicle Dataset (RVD). This dataset contains images taken in different traffic scenarios and weather conditions, which were collected by our autonomous vehicle research group about one year ago.
2. Caltech lane dataset. This dataset contains 1224 tabbed images taken in four different places with the weather condition, however, limited to sunny days.
3. Tusimple benchmark dataset. The Tusimple benchmark dataset contains 6408 labeled images, but again lacks lane boundary data for different traffic or weather conditions.

In the three above-mentioned datasets, RVD contains mainly data of highway or urban roads in China, while most of the other two datasets' data was collected on Americans roads. Except for evaluations of algorithm quantitative tests on these open-source datasets, we have also compared our experiment results with that of other state-of-the-art methods. Besides, we also implemented our proposed algorithm on embedded platforms (i.e., Nvidia TX2) to verify the capability of its real-time performance in real autonomous vehicles, along with algorithm migration and application on a real self-driving car named Pioneer in our research group. Our experiment platform is shown in Fig. 3(b).

4.1 Model Training

The input of our proposed model contains images with a resolution of 1280×1024 pixels. In order to speed up the operation and reduce the computational complexity, a set of pre-processing and transforms of original image input are needed before sending these images to the core CNN network. The final resolution of CNN input image patches is 110×150 pixels. Due to the special structure of our CNN network, there is no pre-training model available, which means that we need to train CNN from scratch. We first train the lane boundary classifier. Next, we change the CNN learning rate, batch size and dropout rate for training the lane boundary regression as well as classification tasks in the same time. After that use the Adaptive Moment Estimation optimization (Adam) and adjust the proportion of regression loss to classification loss to 4:1. We then use the Rectifier neural networks (ReLU) activation function and add the softmax layer to the output of the classification network.

4.2 Evaluation

To estimate our model's effectiveness, we evaluated our algorithm by using precision and recall. Since we did the boundary type classification and lane boundary regression simultaneously, only samples with the correct classification and regression results will be set as True Positive (TP). As for the measurement of the regression result, we set the

threshold to a value which is supposed to be accurate when it is larger than the deviation of the predicted lane boundary coordinates and the ground-true. In the experiment we set the threshold to 5 pixels. Table 2 shows the accuracy of our algorithm in various traffic scenarios.

Road Vehicle Dataset (RVD): Since RVD contains images collected from different weather conditions and traffic scenarios, we can evaluate our methods' robustness on it. There are some visualized results, which are shown in Fig. 4. And also we measured precision and recall which are shown in Table 1. The performance on RVD shows that our method can accurately recognize the lane boundaries in several environments especially in bad weather conditions.

Fig. 4. Visualization result on RVD

Caltech Lane Dataset: We compared our methods with Lee's [9] on Caltech lane dataset. Table 2 shows that our methods outperformed Lee's method in such scenario. Since Caltech lane dataset only contains 1224 labeled images, this result represent that our method can extract the lane boundaries' features from a relative small scale of dataset efficiently.

Tusimple Benchmark Dataset: Tusimple benchmark dataset evaluates the methods performance on highway. We also tested our methods on it. Our methods only achieved about 90% accuracy. Although the images in Tusimple dataset are collected from undulated roads, which seriously impact the quality of the top view image, our methods show the robustness in this situation. Some of the visualized results on Caltech lane dataset and Tusimple benchmark dataset shown in Fig. 4.

4.3 Implementation on Embedded System

Since self-driving platforms have limited computing resources, it is meaningful to implement the lane boundary detection algorithm on the embedded platform.

Table 1. Evaluation on RVD dataset.

Scenario	Total image	Precision [%]	Recall [%]	F_1 [%]
Highway	5066	97.067	95.389	96.221
Rainy and snowy	3802	93.127	94.864	93.987
Night	3566	92.376	91.272	91.821
Urban	3467	98.261	96.246	97.192
Heavy urban	3257	93.491	92.341	92.912
Nonline	1237	95.912	94.366	95.133
Curve	1520	96.491	92.231	94.312
Complex illumination	1382	94.821	93.412	94.112

Table 2. Comparison with Lee's [9] work on Caltech dataset.

Traffic	Methods	Precision [%]	Recall [%]	F_1 [%]
Cordova1	Ours	97.326	96.651	**96.987**
	Lee's	–	–	88.4
Washington1	Ours	98.067	97.129	**97.596**
	Lee's	–	–	86.9

We choose the GPU-based embedded platform Nvidia TX2, which has 256 CUDA cores and low power consumption as the experiment's platform. After implemented and optimized proposed algorithm on Nvidia TX2, the frame rate has achieved 30 FPS on embedded system (Fig. 5).

Fig. 5. Visualization result on Caltech lane dataset and Tusimple benchmark dataset

5 Conclusion and Future Work

In this paper, we proposed a lane detection algorithm based on deep learning and spatial-temporal information. We used the spatial and temporal constraints of a lane to estimate the position of the lane and also applied CNN to predict the lane boundary type and the exact location of a lane. Based on the reorganization of boundary types, we propose a lane detection framework that corresponds to human driving habits, which reasonably reduces the computational complexity of the algorithm and improves the robustness of it. Experiments showed that our algorithm can accurately and quickly detect lane boundary information. In the future work, we hope to combine laser sensor information with our algorithm to further improve the accuracy and robustness of lane boundary inspection.

Acknowledge. This research was partially supported by the National Natural Science Foundation of China (No. 61773312, 61790563), the Programme of Introducing Talents of Discipline to University (No. B13043). We are grateful to the reviewers for taking the time to read this article.

References

1. Aly, M.: Real time detection of lane markers in urban streets. In: Intelligent Vehicles Symposium, pp. 7–12. IEEE (2008)
2. Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J.-A., et al.: End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316 (2016)
3. Chen, S., Zhang, S., Shang, J., Chen, B., Zheng, N.: Brain inspired cognitive model with attention for self-driving cars. arXiv preprint arXiv:1702.05596 (2017)
4. He, B., Ai, R., Yan, Y., Lang, X.: Accurate and robust lane detection based on dual-view convolutional neutral network. In: Intelligent Vehicles Symposium (IV), 2016 IEEE, pp. 1041–1046. IEEE (2016)
5. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
6. Hillel, A.B., Lerner, R., Levi, D., Raz, G.: Recent progress in road and lane detection: a survey. Mach. Vis. Appl. **25**(3), 727–745 (2014)
7. Hoiem, D., Efros, A.A., Hebert, M.: Putting objects in perspective. Int. J. Comput. Vis. **80** (1), 3–15 (2008)
8. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
9. Labayrade, R., Douret, J., Laneurit, J., Chapuis, R.: A reliable and robust lane detection system based on the parallel use of three algorithms for driving safety assistance. IEICE Trans. Inf. Syst. **89**(7), 2092–2100 (2006)
10. Lee, S., Kweon, I.S., Kim, J., Yoon, J.S., Shin, S., Bailo, O., Kim, N., Lee, T.-H., Hong, H. S., Han, S.-H.: VPGNet: Vanishing point guided network for lane and road marking detection and recognition. In 2017 IEEE International Conference on Computer Vision (ICCV), pp. 1965–1973. IEEE (2017)

11. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., Berg, A.C.: SSD: single shot MultiBox detector. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9905, pp. 21–37. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46448-0_2
12. Narote, S.P., Bhujbal, P.N., Narote, A.S., Dhane, D.M.: A review of recent advances in lane detection and departure warning system. Pattern Recogn. **73**, 216–234 (2018)
13. Nieto, M., Salgado, L., Jaureguizar, F., Arróspide, J.: Robust multiple lane road modeling based on perspective analysis. In: 15th IEEE International Conference on Image Processing, ICIP 2008, pp. 2396–2399. IEEE (2008)
14. Pan, X., Shi, J., Luo, P., Wang, X., Tang, X.: Spatial as deep: spatial cnn for traffic scene understanding. arXiv preprint arXiv:1712.06080 (2017)
15. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 779–788 (2016)
16. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems, pp. 91–99 (2015)
17. Samadzadegan, F., Sarafraz, A., Tabibi, M.: Automatic lane detection in image sequences for vision-based navigation purposes. In: ISPRS Image Engineering and Vision Metrology (2006)
18. Satzoda, R.K., Sathyanarayana, S., Srikanthan, T., Sathyanarayana, S.: Hierarchical additive hough transform for lane detection. IEEE Embed. Syst. Lett. **2**(2), 23–26 (2010)
19. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
20. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., et al.: Going deeper with convolutions. In: CVPR (2015)