Proceeding of the IEEE
International Conference on Robotics and Biomimetics
Dali, China, December 2019

Accurate Lane Detection with Atrous Convolution and Spatial Pyramid Pooling for Autonomous Driving

Yuxiang Sun[1], Lujia Wang[2], Yongquan Chen[3], and Ming Liu[1], *Senior Member, IEEE*

***Abstract*— Lane detection is a fundamental capability for autonomous driving. Many effective lane detection algorithms based on traditional computer vision and recent deep learning technologies have been proposed. However, the current state-of-the-art lane detection accuracy is still not satisfactory for realizing fully autonomous driving. Thus, this paper proposes a new lane detection network using atrous convolution and spatial pyramid pooling techniques to improve the lane detection accuracy. We address the detection problem with pixel-wise semantic segmentation. Our network consists of one encoder and two decoders, which outputs a binary segmentation map and an embedded feature map, respectively. The embedded feature map is employed for clustering algorithms to separate segmented lane pixels into different lanes. The experimental results on the public Tusimple dataset show that our network outperforms the state-of-the-arts.**

I. Introduction

Lane detection is important for autonomous driving. It is a fundamental component for many autonomous driving tasks, such as Lane Departure Warning Systems (LDWS) [1] and Lane Keeping Assistance Systems (LKAS) [2]. In addition, accurate lane detection could also benefit ego-vehicle localization and navigation [3]–[11], which is crucial for autonomous vehicles to make high-level decisions, such as route planning and collision avoidance. Lane detection has attracted great attentions from both the academia and industry. Many effective lane detection algorithms have been developed in the past decade [12]. There are algorithms that are even incorporated into commercial Advanced Driver Assistant System (ADAS) [13] products.

Generally, existing lane detection algorithms could be divided into two classes: one class of methods are based on traditional computer vision; and the other class of methods are based on recent deep learning technologies. The traditional methods usually adopt the detection and tracking pipeline [14]. In the detection stage, an image is firstly cropped with a region of interest, and edge detection algorithms are applied on the interested regions to extract edge pixels. Then, the pixels on the extracted edges are used to detect lanes with line fitting algorithms, such as Hough transform [15]. In the tracking stage, tracking algorithms, such as Kalman Filter or Particle Filter [16], are used to increase the robustness of the lane detection.

The development of deep learning technologies has virtually changed the landscape in computer vision [17], as well as robotics [18] and intelligent transportation systems [19]. Convolutional neural networks (CNN) [20] have achieved great success in vision-based detection and segmentation applications [21]–[28]. Researchers in lane detection also resort to using deep learning technologies to improve the detection accuracy. Some effective networks, such as LaneNet [29], have been developed. Generally, the results of using deep learning have greatly outperformed those using traditional computer vision algorithms in terms of accuracy and robustness. Despite the success of using deep learning technologies, the current state-of-the-art performance is still not satisfactory for practical fully autonomous driving. In order to improve the performance of lane detection, this paper proposes to use atrous convolution and spatial pyramid pooling techniques. We treat lane detection as a semantic segmentation problem and build our network based on LaneNet.

The main novelty of this paper is the use of atrous convolution and spatial pyramid pooling techniques for pixel-wise lane detection. We summarize the contributions of this paper as follows:

1) We develop a new network based on LaneNet [29] for lane detection using atrous convolution and spatial pyramid pooling techniques.
2) We demonstrate that our network outperforms the state-of-the-arts on the public Tusimple dataset.

The remainder of this letter is structured as follows. In section II, related work has been reviewed. In section III, we describe our network in detail. Section IV presents the experimental results and discussions. The conclusion and future work are drawn in the last section.

II. Related Work

Since we treat the lane detection problem as a semantic segmentation problem. The related work to this paper includes deep learning-based semantic segmentation, and lane detection using semantic segmentation technologies. In this section, we review several representative work in the two research fields.

A. Semantic Segmentation Networks

The first work that solves semantic segmentation with CNN in an end-to-end fashion is the Fully Convolutional Networks (FCNs) proposed by Shelhamer *et al.* [30]. They

[1]Yuxiang Sun and Ming Liu are with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong SAR, China. (email: eeyxsun@ust.hk, sun.yuxiang@outlook.com; eelium@ust.hk).

[2]Lujia Wang is with the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China. (email: lj.wang1@siat.ac.cn, rugga.wang@gmail.com).

[3]Yongquan Chen is with the Shenzhen Institute of Artificial Intelligence and Robotics for Society, Institute of Robotics and Intelligent Manufacturing, The Chinese University of Hong Kong, Shenzhen, China. (email: yqchen@cuhk.edu.cn).

978-1-7281-6321-5/19//$31.00 ©2019 IEEE

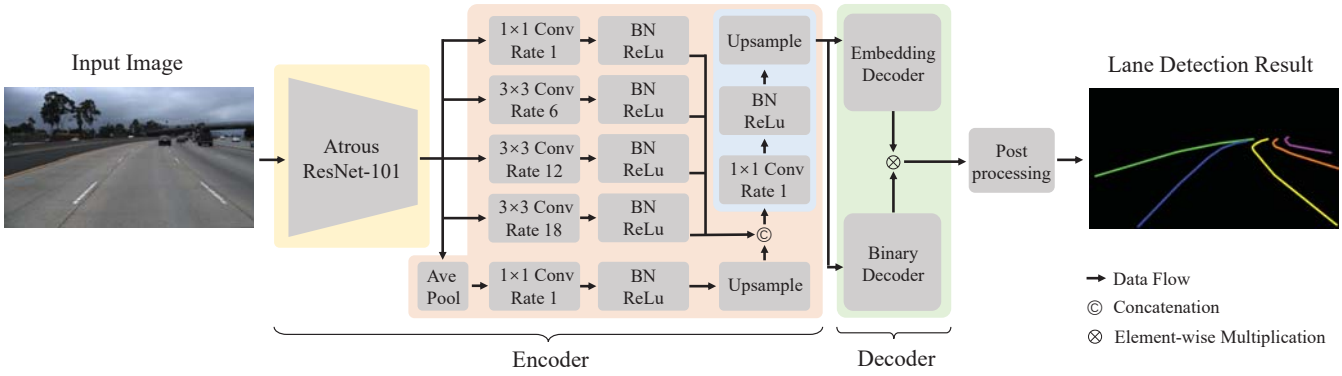


Fig. 1: The structure of our instance segmentation network for lane detection. The yellow box represents the Atrous CNN module. In this paper, we use the Atrous ResNet-101. The pink and blue boxes represent the Atrous SPP module. The decoder module is shown in the green box. *Conv*, *BN*, *Ave Pool* represent convolutional, batch normalization and average pooling layers, respectively. *Rate* mean the dilation rates for the *Conv* layers. In the detection result, different lanes are denoted with different colors. The figure is best viewed in color.

employed image classification networks, such as VGG-16 [31] or GoogLeNet [32], to extract visual features from input images. The fully connected layers in the networks are removed. Noh *et al.* [33] proposed a semantic segmentation network named DeconvNet, which included a convolutional module for feature extraction and a deconvolutional module for resolution restoration. The convolutional module employed VGG-16 [31] as the backbone. The deconvolutional module was designed as a mirrored version of the convolutional module. Badrinarayanan *et al.* [34] firstly introduced the Encoder-Decoder concept in SegNet for semantic segmentation. The encoder was designed to extract visual features and the decoder was designed to restore the feature map resolution. The Encoder-Decoder fashion was widely used in later semantic segmentation networks. Ronneberger *et al.* [35] developed UNet for semantic segmentation in the biomedical imaging [36] research field. They introduced the U-shape structure that was similar to the Encoder-Decoder structure. The U-shape structure connected the feature extraction path and the resolution restoration path, which was believed to be able to reduce the loss of spatial information of feature maps. Zhao *et al.* [37] developed PSPNet for semantic segmentation by incorporating context information in a scene to improve the segmentation performance. They designed a pyramid pooling module to retrieve local and global context information at different receptive-field levels. Yu *et al.* [38] re-thought semantic segmentation in a macroscopic point of view. They believed that the intra-class inconsistency and inter-class indistinction led to the inferior performance of semantic segmentation. Thus, they proposed the Discriminative Feature Network (DFN) that contains a smooth network and a boarder network to respectively address the two issues.

B. Lane Detection using Semantic Segmentation

Pan *et al.* [39] proposed Spatial CNN (SCNN) which propagates spatial information through modified CNN structures. SCNN outperformed CNN especially in long thin structure, like poles or traffic lanes. For lane detection, in order to discriminate segmented pixels into different lanes, they designed a small network taking as input the probability maps, and giving the output of a vector with 4-elements ranging from 0 to 1 to indicate the probability for each lane. Neven *et al.* [29] addressed lane detection as an instance segmentation problem. Each lane corresponds to an instance. They proposed LaneNet that consists of one encoder and two decoders. The two decoders output a binary segmentation map and an embedding feature map, respectively. The binary segmentation map is trained via the standard cross entropy loss, while the embedding feature map is trained with specially designed instance segmentation losses. The binary segmentation map and the embedding feature map are combined together, and then sent to a clustering algorithm to get the final instance segmentation result.

TABLE I: The detailed configurations for the convolutional layers in the Atrous SPP network that is shown in Fig. 1. *Conv* represents convolutional layer. The convolutional layers from top to bottom in the pink box of Fig. 1 are numbered with 1-5, respectively. The convolutional layer in the blue box of Fig. 1 is numbered with 6.

Conv No.	Input Channel	Output Channel	Kernel Size	Stride	Padding	Dilation Rate
1	2048	256	1×1	1	0	1
2	2048	256	3×3	1	6	6
3	2048	256	3×3	1	12	12
4	2048	256	3×3	1	18	18
5	2048	256	1×1	1	0	1
6	1280	256	1×1	1	0	1

III. The Proposed Method

A. Network Overview

The structure of our network is displayed in Fig. 1. The network is built on LaneNet [29]. It mainly consists of one

TABLE II: The detailed configurations for the *TransConv* and *Conv* layers in the decoders. *TransConv* means transposed convolutional layer. The numbers of the output channel of the last transposed convolutional layer is 4 for the *Embedding Decoder* and 2 for the *Binary Decoder*, which are shown as $4/2$ in the table.

Layer Name	Input Channel	Output Channel	Kernel Size	Stride
TransConv 1	512	256	2×2	2
Conv	256	128	1×1	1
TransConv 2	128	$4/2$	4×4	4

encoder and two decoders. The two decoders are respectively named as the *Embedding Decoder* and *Binary Decoder*. They output the embedding feature map and binary segmentation map, respectively. The embedding feature map and binary segmentation map are element-wisely multiplied, and then sent to a post-processing module to get the instance segmentation for the lane detection. Different lanes are labelled with different numbers, which are depicted with different colors in Fig. 1.

B. The Encoder

We replace the original encoder of LaneNet with the sequential combination of the Atrous ResNet-101 and the Spatial Pyramid Pooling (SPP) networks, which are proposed in DeepLab v3+ [40]. As shown in Fig. 1, there are 5 branches in the SPP network, among which 4 branches consist of a *Conv-BN-ReLu* module and 1 branch consists of an *AvePool-Conv-BN-ReLu* module. The feature maps from the 5 branches are concatenated together and sent to a *Conv* layer for feature fusion. The detailed configurations for the convolutional layers in the Atrous SPP network are displayed in Tab. I. The 5 branches retrieve the spatial information of feature maps at different receptive filed scales. Note that the *Ave Pool* module in the bottom branch resizes the resolution of the feature map to 1×1, which encodes the global feature. In order to concatenate the feature map from the bottom branch with those from the other 4 branches, an interpolation upsample operation is employed in the bottom branch. As shown in Tab. I, all the strides of convolutional layers in the Atrous SPP network is 1, indicating that the convolutional layers keep the resolution of feature maps unchanged.

C. The Decoders

The structures of the *Embedding Decoder* and *Binary Decoder* are identical to each other, except the number of output dimension. The numbers of the output dimension of the *Embedding Decoder* and *Binary Decoder* are 4 and 2, respectively. The *Embedding Decoder* transforms feature maps to 4-element embedding vector maps. As the binary decoder gives the binary decision (background or lane), the number of the output channel is set to 2. The decoders sequentially consist of a *TransConv-BN-ReLu* module, a *Conv-BN-ReLu* module and a TransConv layer. *TransConv* means transposed convolutional layer. We number the transposed convolutional layer in the TransConv-BN-ReLu module and the last transposed convolutional layer with 1 and 2, respectively. The detailed configurations for the convolutional and transposed convolutional layers of the decoders are listed in Tab. II. As we can see, the transposed convolutional layers reduce the dimensions and upsample the feature maps to the desired resolution, while the convolutional layer only reduces the dimensions.

D. The Loss Functions

To learn the embedding feature map, we directly adopt the loss functions that are proposed in [29]. The loss functions realize the instance segmentation for the lanes. They are the variance loss ℓ_{var} and distance loss ℓ_{dist}. The variance loss ℓ_{var} enforces the minimization of the distances of the pixel embeddings in the same lane. The distance loss ℓ_{dist} enforces the maximization of the distances of the pixel embeddings in different lanes. For the binary segmentation, we use the standard cross entropy loss ℓ_{bin} for the output of the *Binary Decoder*. The total loss ℓ is computed as:

$$\ell = \ell_{bin} + \alpha\ell_{var} + \beta\ell_{dist}, \tag{1}$$

where α and β are weighting parameters for the two losses ℓ_{var} and ℓ_{dist}. In this paper, we set both α and β to 1.0. The loss ℓ_{var} is computed as:

$$\ell_{var} = \frac{1}{C}\sum_{c=1}^{C}\frac{1}{N_c}\sum_{i=1}^{N_c}\phi^2\big[\mathcal{D}(\mu_c, p_i) - \lambda_{var}\big], \tag{2}$$

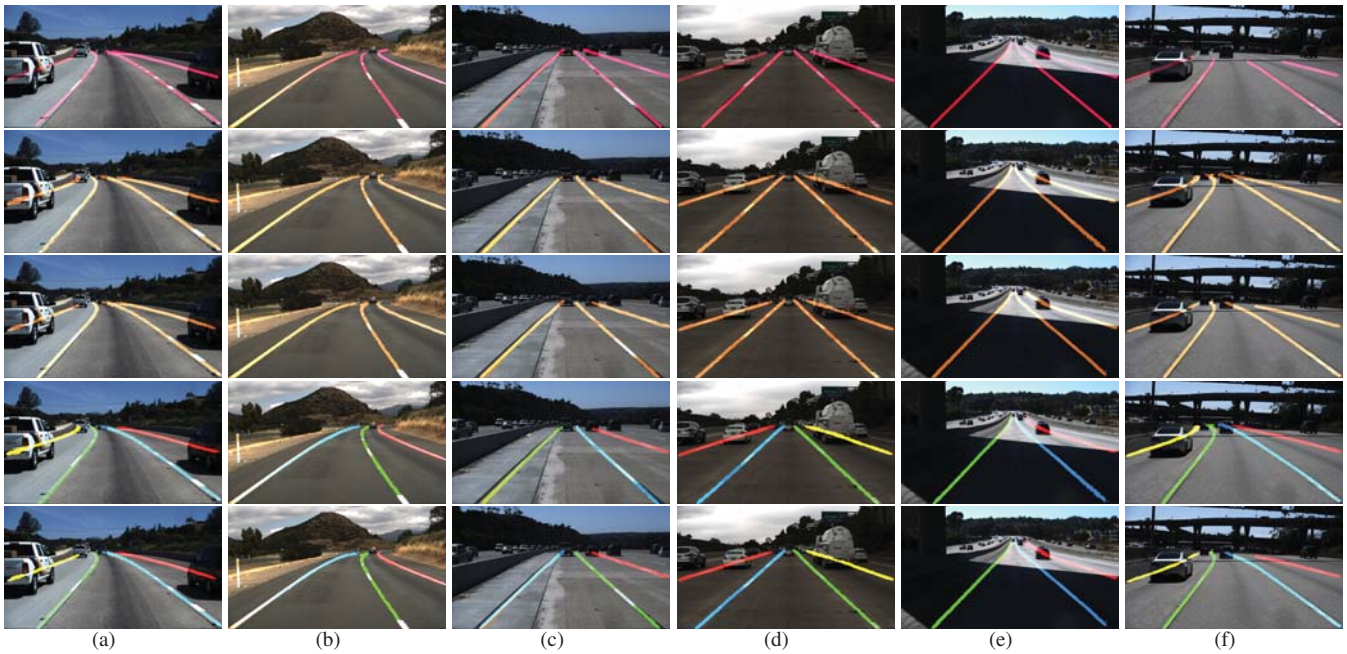
where C is the number of clusters (lanes), N_c is the number of pixel embeddings in a cluster, C and N_c come from ground truth, $\phi(\cdot)$ is the ReLu function, $\mathcal{D}(\cdot)$ calculates the 2-norm distance for the two input vectors, μ_c and p_i are the mean value of the pixel embeddings and the i-th pixel embedding in the cluster c, respectively, λ_{var} is a constant parameter, here we set $\lambda_{var} = 0.5$. The loss ℓ_{dist} is computed as:

$$\ell_{dist} = \frac{1}{C(C-1)}\sum_{c_A=1}^{C}\sum_{c_B=1}^{C}\phi^2\big[\lambda_{dist} - \mathcal{D}(\mu_{c_A}, \mu_{c_B})\big], \tag{3}$$

where c_A and c_B are two clusters, μ_{c_A}, μ_{c_B} are the mean values of the pixel embeddings of cluster c_A and c_B, respectively, λ_{dist} is a constant parameter, here we set $\lambda_{dist} = 3.0$.

E. Post-processing

The embedding feature map is element-wisely multiplied with the binary segmentation map. The resulting feature map is then sent to the post-processing module. The post-processing module mainly consists of a clustering algorithm to cluster the pixels to different lanes according to the values of the embedding vectors. Here, we use the mean-shift clustering algorithm [41]. Post-processing could also include line fitting algorithms to derive mathematical formulas for each lane. However, it is out of the scope of this paper, we refer readers to [29] for more information.



(a) (b) (c) (d) (e) (f)

Fig. 2: Sample acceptable qualitative results from the *test* set. The rows from top to bottom are ground truth binary segmentation results, LaneNet binary segmentation results, our binary segmentation results, LaneNet instance segmentation results and our instance segmentation results. The figure is best viewed in color.

TABLE III: The details of the dataset split scheme. We split the Tusimple dataset into *train* set, *validation* set and *test* set, respectively.

Split	File Name (.json)	Number of Images
Train	label_data_0313	2858
	label_data_0601	410
Validation	label_data_0531	358
Test	test_label	2782

TABLE IV: The detailed configurations for the SGD optimizer used for our training. *LR* represents learning rate. *Decay* means the weight decay rate.

	Initial LR	Momentum	Dampening	Decay	Nesterov
Value	0.01	0.9	0	0.001	TRUE

IV. Experimental Results and Discussions

A. Experimental Setup

1) The Dataset: We use the public Tusimple lane detection dataset[1] in this paper. There are several *json* files that contain the ground truth of lane pixel coordinates. We generate ground truth instance segmentation maps by drawing lines through the pixel coordinates of each lane. The lines are with 15-pixel thickness. Different lanes are annotated with different labels. We split the dataset into *train* set, *validation* set and *test* set, which respectively include 3268, 358 and 2782 images. The details about the dataset split are listed in Tab. III. Note that all the experimental results in this paper are obtained with the *test* set.

2) Training Details: Our network is implemented with PyTorch 1.2, CUDA 10.0 and cuDNN 7.0 on Ubuntu 18.04. We train the models on a PC with an Intel i7 CPU and an NVIDIA GeForce GTX 1080 Ti graphics card. As the graphics memories of the card are limited to 11 GB, we adjust the batch sizes to fit the graphics memories accordingly for each network. In order to increase the training efficiency, we resize all the images from the resolution of 720×1280 to 288×512. We also normalize all the images with the mean $(0.485, 0.456, 0.406)$ and standard deviation $(0.229, 0.224, 0.225)$ for the three image channels. We employ the Stochastic Gradient Descent (SGD) optimization solver to train the networks. The detailed parameters are listed in Tab. IV. The learning rate is decayed during the training process. We train the networks until the validation loss converges.

[1]https://github.com/TuSimple/tusimple-benchmark/wiki

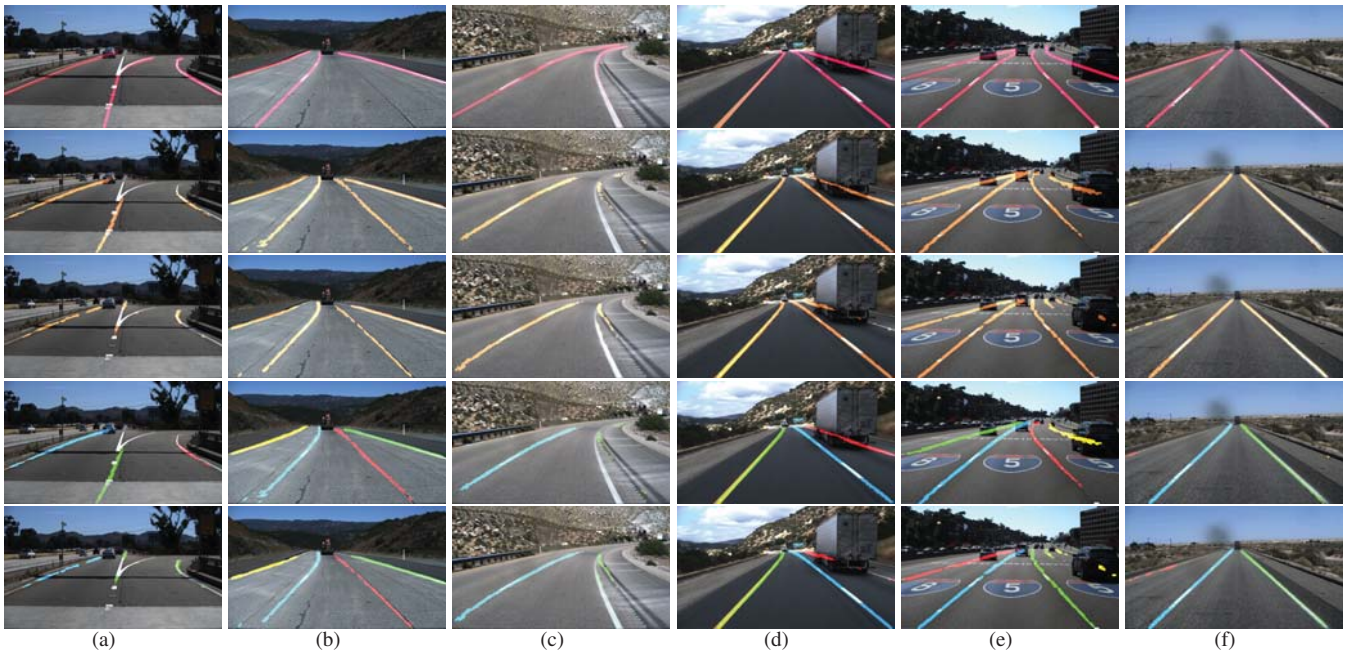
B. Evaluation Metrics

We quantitatively evaluate the binary segmentation results using the metrics Precision (Pre), Recall (Rec) and IoU. They are computed with the formulas:

$$\text{Pre} = \frac{\sum_{n=1}^{N} \text{TP}_n}{\sum_{n=1}^{N} \text{TP}_n + \sum_{n=1}^{N} \text{FP}_n}, \tag{4}$$

$$\text{Rec} = \frac{\sum_{n=1}^{N} \text{TP}_n}{\sum_{n=1}^{N} \text{TP}_n + \sum_{n=1}^{N} \text{FN}_n}, \tag{5}$$

$$\text{IoU} = \frac{\sum_{n=1}^{N} \text{TP}_n}{\sum_{n=1}^{N} \text{TP}_n + \sum_{n=1}^{N} \text{FN}_n + \sum_{n=1}^{N} \text{FP}_n}, \tag{6}$$



(a) (b) (c) (d) (e) (f)

Fig. 3: Sample unacceptable qualitative results from the *test* set. The rows from top to bottom are ground truth binary segmentation results, LaneNet binary segmentation results, our binary segmentation results, LaneNet instance segmentation results and our instance segmentation results. The figure is best viewed in color.

TABLE V: The quantitative evaluation results for LaneNet and our network. Larger values indicate better performance for *Pre*, *Rec* and *IoU*, which are annotated with ↑. Smaller values indicate better performance for the time cost, which is annotated with ↓.

	Pre ↑	Rec ↑	IoU ↑	Time Cost (ms) ↓
SCNN [39]	0.2836	**0.7114**	0.2543	17.88
LaneNet [29]	0.2742	0.6657	0.2410	18.96
Ours	**0.3054**	0.6796	**0.2670**	**13.03**

where *TP*, *FN* and *FP* respectively represent the number of pixels for the True Positives, False Negatives and False Positives [42]. Specifically, *TP* is the number of lane pixels that are correctly detected as lanes, *FN* is the number of lane pixels that are wrongly detected as background, and *FP* is the number of background pixels that are wrongly detected as lanes. N is number of tested images, here we have $N = 2782$. n is the frame index.

C. Comparative Results

We compare our network with SCNN [39] and LaneNet [29]. To get binary segmentation results for SCNN, we add a sequential block after the layer No.2 of SCNN. The block sequentially consists of an interpolation up-sampling function and a 1×1 convolutional layer, which are used to increase the feature-map resolution to the input resolution and reduce the dimensionality to 2. Tab. V displays the quantitative comparative results. The *Time Cost* in the table represents the average forward time cost per image on the NVIDIA GeForce GTX 1080 Ti graphics card. As we can see, our network achieves the superior performance.

Fig. 2 show some acceptable results for LaneNet and our network. For the binary segmentation, we use the single color to annotate the detected lanes. For the instance segmentation, we use different colors to annotate different lanes. We can see from Fig. 2 that both the networks are able to robustly detect lanes in these scenes. In Fig. 2 (a) and (d-f), the networks are able detect lanes even there are occlusions. Fig. 2 (e) shows the robustness of the networks to the varying lighting condition. Fig. 3 show some unacceptable results. In Fig. 3 (a), some lane pixels are not correctly segmented. In Fig. 3 (b) and (c), the road crack and road curb are wrongly identified as lanes, respectively. In Fig. 3 (d) and (e), lanes are not robustly detected due to the occlusions from other vehicles. In Fig. 3 (f), the lane near the road curb is not correctly detected.

V. Conclusion

We presented here a lane detection network that uses atrous convolution and spatial pyramid pooling for autonomous driving. Our network is built on LaneNet. The experimental results on the public Tusimple dataset show that our network outperforms the SCNN and LaneNet. The future work of this paper includes the robustness enhancement on challenging scenarios, such as road crack or occlusions. In addition, line fitting networks will be integrated with our network to achieve end-to-end learning of the mathematical formulas of lane lines. To thoroughly evaluate our network, we will also record and manually label our own lane detection dataset under various lighting and weather conditions.

REFERENCES

[1] S. P. Narote, P. N. Bhujbal, A. S. Narote, and D. M. Dhane, "A review of recent advances in lane detection and departure warning system," *Pattern Recognition*, vol. 73, pp. 216–234, 2018.

[2] K. Lee, S. E. Li, and D. Kum, "Synthesis of Robust Lane Keeping Systems: Impact of Controller and Design Parameters on System Performance," *IEEE Transactions on Intelligent Transportation Systems*, 2018.

[3] Y. Sun, M. Liu, and M. Q.-H. Meng, "Motion removal for reliable RGB-D SLAM in dynamic environments," *Robotics and Autonomous Systems*, vol. 108, pp. 115 – 128, 2018.

[4] G. Wan, X. Yang, R. Cai, H. Li, Y. Zhou, H. Wang, and S. Song, "Robust and precise vehicle localization based on multi-sensor fusion in diverse city scenes," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 4670–4677.

[5] D. Zhu, T. Li, D. Ho, C. Wang, and M. Q.-H. Meng, "Deep Reinforcement Learning Supervised Autonomous Exploration in Office Environments," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7548–7555, 2018.

[6] Y. Yao, M. Xu, C. Choi, D. J. Crandall, E. M. Atkins, and B. Dariush, "Egocentric vision-based future vehicle localization for intelligent driving assistance systems," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 9711–9717.

[7] S. Kuutti, S. Fallah, K. Katsaros, M. Dianati, F. Mccullough, and A. Mouzakitis, "A survey of the state-of-the-art localization techniques and their potentials for autonomous vehicle applications," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 829–846, 2018.

[8] P. Cai, Y. Sun, Y. Chen, and M. Liu, "Vision-Based Trajectory Planning via Imitation Learning for Autonomous Vehicles," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, Oct 2019, pp. 2736–2742.

[9] H. Huang, Y. Sun, and M. Liu, "Reliable Monocular Ego-Motion Estimation System in Rainy Urban Environments," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, Oct 2019, pp. 1290–1297.

[10] C. Wang, W. Chi, Y. Sun, and M. Q.-H. Meng, "Autonomous Robotic Exploration by Incremental Road Map Construction," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 4, pp. 1720–1731, Oct 2019.

[11] J. Cheng, Y. Sun, and M. Q.-H. Meng, "Improving monocular visual slam in dynamic environments: an optical-flow-based approach," *Advanced Robotics*, vol. 33, no. 12, pp. 576–589, 2019.

[12] Y. Xing, C. Lv, L. Chen, H. Wang, H. Wang, D. Cao, E. Velenis, and F.-Y. Wang, "Advances in vision-based lane detection: algorithms, integration, assessment, and perspectives on ACP-based parallel vision," *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 3, pp. 645–661, 2018.

[13] K. Bengler, K. Dietmayer, B. Farber, M. Maurer, C. Stiller, and H. Winner, "Three decades of driver assistance systems: Review and future perspectives," *IEEE Intelligent transportation systems magazine*, vol. 6, no. 4, pp. 6–22, 2014.

[14] C. Lee and J.-H. Moon, "Robust lane detection and tracking for real-time applications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 12, pp. 4043–4048, 2018.

[15] J. Niu, J. Lu, M. Xu, P. Lv, and X. Zhao, "Robust lane detection using two-stage feature extraction with curve fitting," *Pattern Recognition*, vol. 59, pp. 225–233, 2016.

[16] Y. Sun, M. Liu, and M. Q.-H. Meng, "Improving RGB-D SLAM in dynamic environments: A motion removal approach," *Robotics and Autonomous Systems*, vol. 89, pp. 110 – 122, 2017.

[17] Y. Sun, W. Zuo, and M. Liu, "RTFNet: RGB-Thermal Fusion Network for Semantic Segmentation of Urban Scenes," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2576–2583, July 2019.

[18] H. Wang, Y. Sun, and M. Liu, "Self-Supervised Drivable Area and Road Anomaly Segmentation Using RGB-D Data For Robotic Wheelchairs," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4386–4393, Oct 2019.

[19] H. Wang, Y. Yu, Y. Cai, X. Chen, L. Chen, and Q. Liu, "A Comparative Study of State-of-the-Art Deep Learning Algorithms for Vehicle Detection," *IEEE Intelligent Transportation Systems Magazine*, vol. 11, no. 2, pp. 82–95, 2019.

[20] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai *et al.*, "Recent advances in convolutional neural networks," *Pattern Recognition*, vol. 77, pp. 354–377, 2018.

[21] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3d object detection from rgb-d data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 918–927.

[22] D. Zhu, T. Li, D. Ho, T. Zhou, and M. Q.-H. Meng, "A Novel OCR-RCNN for Elevator Button Recognition," *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3626–3631, 2018.

[23] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4490–4499.

[24] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE transactions on neural networks and learning systems*, 2019.

[25] L. Maczyta, P. Bouthemy, and O. Le Meur, "CNN-based temporal detection of motion saliency in videos," *Pattern Recognition Letters*, 2019.

[26] J. Cheng, Y. Sun, and M. Q.-H. Meng, "Robust Semantic Mapping in Challenging Environments," *Robotica*, pp. 1–15, 2019.

[27] F. Lateef and Y. Ruichek, "Survey on semantic segmentation using deep learning techniques," *Neurocomputing*, vol. 338, pp. 321–348, 2019.

[28] G. L. Oliveira, C. Bollen, W. Burgard, and T. Brox, "Efficient and robust deep networks for semantic segmentation," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 472–491, 2018.

[29] D. Neven, B. D. Brabandere, S. Georgoulis, M. Proesmans, and L. V. Gool, "Towards End-to-End Lane Detection: an Instance Segmentation Approach," *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 286–291, 2018.

[30] E. Shelhamer and J. Long and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 640–651, April 2017.

[31] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[32] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 1–9.

[33] H. Noh, S. Hong, and B. Han, "Learning Deconvolution Network for Semantic Segmentation," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, pp. 1520–1528.

[34] V. Badrinarayanan and A. Kendall and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, Dec 2017.

[35] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Cham: Springer International Publishing, 2015, pp. 234–241.

[36] T. Falk, D. Mai, R. Bensch, Ö. Çiçek, A. Abdulkadir, Y. Marrakchi, A. Böhm, J. Deubner, Z. Jäckel, K. Seiwald *et al.*, "U-Net: deep learning for cell counting, detection, and morphometry," *Nature methods*, vol. 16, no. 1, p. 67, 2019.

[37] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid Scene Parsing Network," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 6230–6239.

[38] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "Learning a Discriminative Feature Network for Semantic Segmentation," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018, pp. 1857–1866.

[39] X. Pan, J. Shi, P. Luo, X. Wang, and X. Tang, "Spatial as deep: Spatial cnn for traffic scene understanding," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[40] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 801–818.

[41] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 5, pp. 603–619, 2002.

[42] Y. Sun, M. Liu, and M. Q.-H. Meng, "Active Perception for Foreground Segmentation: An RGB-D Data-Based Background Modeling Method," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 4, pp. 1596–1609, Oct 2019.