Row-wise LiDAR Lane Detection Network with Lane Correlation Refinement

Dong-Hee Paek[1], Kevin Tirta Wijaya[2], and Seung-Hyun Kong*[1]

***Abstract*— Lane detection is one of the most important functions for autonomous driving. In recent years, deep learning-based lane detection networks with RGB camera images have shown promising performance. However, camera-based methods are inherently vulnerable to adverse lighting conditions such as poor or dazzling lighting. Unlike camera, LiDAR sensor is robust to the lighting conditions. In this work, we propose a novel two-stage LiDAR lane detection network with row-wise detection approach. The first-stage network produces lane proposals through a global feature correlator backbone and a row-wise detection head. Meanwhile, the second-stage network refines the feature map of the first-stage network via attention-based mechanism between the local features around the lane proposals, and outputs a set of new lane proposals. Experimental results on the K-Lane dataset show that the proposed network advances the state-of-the-art in terms of F1-score with 30% less GFLOPs. In addition, the second-stage network is found to be especially robust to lane occlusions, thus, demonstrating the robustness of the proposed network for driving in crowded environments.**

I. INTRODUCTION

To be able to navigate from a point to another, an autonomous driving agent needs to plan a safe and efficient route according to the environmental conditions. Therefore, the ability of perceiving the environment through raw sensor measurements is crucial for the autonomous driving. One of the perception tasks for autonomous driving is the lane detection task, where the autonomous driving agent needs to detect the location of lane lines on the roads.

Extensive studies have been conducted on the lane detection task, particularly with the RGB camera sensors. In the earlier days, rule-based and heuristic systems have been developed to provide lane detection capability in limited predefined environments [1][2][3]. Recently, data-driven approaches become popular owing to the advancements of deep learning. Various neural networks for camera-based lane detection have been developed [4][5][6], with promising accuracy in most of driving conditions.

However, RGB camera has an inherent weakness towards harsh lighting conditions such as low or dazzling light. This is evident in the widely-used CULane benchmark [7], where the performance degradation of various camera-based lane detection networks occurs in the dazzling light and dark. As an autonomous driving agent needs to be robust in various driving conditions, the vulnerability of the existing camera-based lane detection methods towards adverse lighting conditions need to be resolved.

One possible solution to the problem is to use Light Detection and Ranging (LiDAR) sensor. Since the LiDAR sensor emits infrared signals, which is hardly interfered by the visible light, adverse lighting conditions do not affect its measurement capability significantly. Moreover, unlike camera images, the LiDAR point cloud does not require a bird's eye view (BEV) projection for motion planning, which often causes lane line distortions.

Despite the several advantage of the LiDAR sensor, only a handful of studies have proposed deep learning-based LiDAR lane detection methods. This is largely due to the absence of publicly-available LiDAR lane detection datasets. As seen in the deep camera-based lane detection field, the majority of the lane detection networks are developed after the publication of open lane detection datasets such as [7][8].

Recently, [9] opens a large-scale LiDAR lane detection dataset, K-Lane, to the public, along with a segmentation-based LiDAR lane detection network (LLDN) baseline. The baseline consists of a projection network, a global feature correlator (GFC), and a segmentation head, which generates a feature map in the BEV format, extracting features via global correlation, and predicting lanes per grid, respectively. While a segmentation-based lane detection network is capable of producing lane detection of various shapes, it is computationally expensive due to the need of processing each grid of the final feature map through shared multi-layer perceptron (MLP).

In this work, we propose a novel LiDAR lane detection network that is computationally efficient and especially effective to the severe occlusion cases. We set the lane detection problem as a row-wise prediction task instead of a semantic segmentation task, so that the detection head of the network performs row-wise MLP operations instead of grid-wise MLP operations. The row-wise formulation leads to a significantly less computational cost than the prior work [9], with about 30% less GFLOPs.

Furthermore, we design an additional second-stage network that refines the output feature map of the first-stage network via correlation of features around the lane proposals. The correlation process is implemented with the dot-product attention mechanism, which allows the network to exchange information between the features of the lane proposals globally. As the lane lines often have some degree of regularity in terms of shapes and distances between each

*corresponding author

[1]Dong-Hee Paek and Seung-Hyun Kong are with the CCS Graduate School of Mobility, Korea Advanced Institute of Science and Technology, 193, Munji-ro, Yuseong-gu, Daejeon, Republic of Korea {donghee.paek, skong}@kaist.ac.kr

[2]Kevin Tirta Wijaya is with the Robotics Program, Korea Advanced Institute of Science and Technology, 291, Daehak-ro, Yuseong-gu, Daejeon, Republic of Korea kevin.tirta@kaist.ac.kr

other, such global correlation process should be advantageous, for example, in detecting lane lines that are occluded by neighboring vehicles. The row-wise two-stage approach enables our proposed network to achieve the state-of-the-art (SOTA) performance with less computational cost compared to the existing baselines.

In a summary, our contributions are as follows:

- We propose a new technique in LiDAR lane detection through row-wise lane prediction. This technique is more computationally efficient compared to existing segmentation-based LiDAR lane detection techniques.
- We design a two-stage LiDAR lane detection network that first predicts lane proposals, and then refines the first-stage feature map via attention-based mechanism between the lane proposals. The refined feature map is then used to predict the final lane detection proposals.
- We demonstrate the excellent performance of the proposed network; the proposed network achieves SOTA performance in K-Lane with an overall F1-score of 82.74% reducing the GFLOPs by about 30%. While the proposed network achieves a slight overall F1-score improvement (i.e., 0.62%) over the prior SOTA network, the proposed network largely improves the F1-score (i.e., 3.24%) for the severe occlusion cases. Thus, the proposed network enables the robust lane detection required for safe autonomous driving in congested traffics where performance degradation has previously been significant.

The rest of this paper is organized as follows: Section II introduces existing works related to our study, Section III details our proposed LiDAR lane detection network, Section IV discusses the experimental results on the K-Lane dataset, and Section V draws conclusion this study.

II. Related Works

In this section, we discuss existing works that are related to our study. We start with a general review of the more matured camera-based lane detection. Then we discuss further on the row-wise lane detection methods. Finally, we cover the existing LiDAR lane detection methods.

A. Camera-based Lane Detection

Traditional camera-based lane detection methods heavily rely on rule-based systems that require various predefined variables such as intensity thresholding [1][2][3]. As the deep learning field becomes more matured, various camera lane detection networks emerge, with promising accuracy in various driving conditions. Most deep learning-based camera lane detection networks utilize convolutional neural network (CNN) as their backbone feature extractor, and a task-specific detection head.

In [10], the detection head is designed to perform anchor-based lane predictions and coordinate offset predictions. In [4], the detection head produce two affinity-field maps that represent the locations of lane lines. In [6], the detection head is conditioned to predict row-wise lane proposals before being further processed by a post-processing algorithm. Compared to other approaches, row-wise lane detection often perform faster while maintaining good accuracy. As such, we design our LiDAR lane detection network with a row-wise paradigm.

B. Row-wise Lane Detection

Row-wise lane detection is proposed by [11], in which lane lines are detected through predicting the lane location probability of each row. That is, for each row, the location of the lane line is determined as the column of which the lane probability is the highest. Unlike segmentation-based lane detection, row-wise lane detection is based on geometric prior which picks only one location per each lane, so that this method may robust to false alarm. Furthermore, [4] modify the argmax operation into taking the sums of each index of the column weighted by its lane probability to enable the gradients to flow through the lane structure loss.

C. LiDAR Lane Detection

In the early LiDAR lane detection methods, lane lines are detected through an intensity thresholding operation with additional heuristics. These methods often incorporate additional algorithms such as Kalman Filter [12], polar coordinates operation [13], or clustering with DBSCAN [14]. However, heuristic methods are not adaptive towards diverse driving conditions due to the need for predefining numerous thershold variables.

More recently, several studies are starting to incorporate deep learning to their LiDAR lane detection methods. In [15], the front view camera images are combined with the 2D BEV images from the LiDAR point cloud to improve the lane detection performance. In [16], a CNN backbone is utilized to detect the ego lane lines in the highways. In [9], a segmentation-based neural network with global feature correlator backbone is used to perform LiDAR lane detection under various environments in the K-Lane dataset.

III. Method

In this section, we describe the proposed network in details. Firstly, we provide an overview of the structure of the proposed network. Then, we explain in details about the components of the network: the feature extractor, the row-wise detection head, and the refinement head. Finally, we express the loss function that is used to train the network.

A. Two-stage Row-wise Lane Detection Network

As shown in Fig. 1, we design the proposed lane detection network in two-stage detection network with a row-wise approach. First, a feature extractor backbone takes the raw point cloud data as an input. The feature extractor then encodes the raw point cloud into a pseudo bird-eye-view (BEV) image, which is further processed by a global feature correlator to produce the output feature map. This output feature map is utilized by the first stage row-wise detection head to predict two values: row-wise existence and row-wise probability. Through the row-wise existence and location probability predictions, we can obtain the first-stage lane detection proposals.

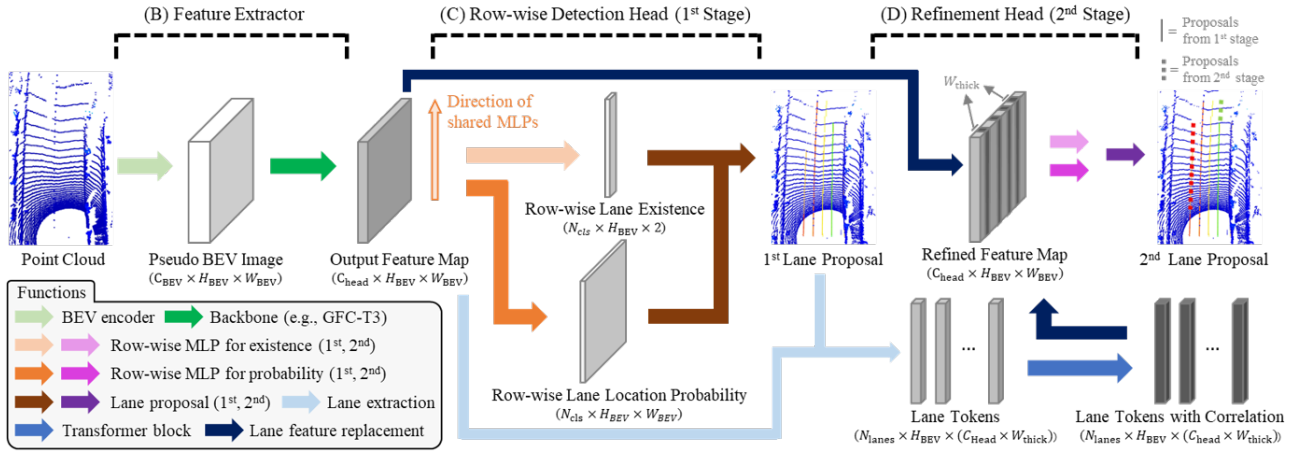

Fig. 1. Overall structure of the proposed network.

To further improve the lane detection accuracy, we use a second-stage refinement head. The refinement head collects features around the location of the lane proposals, which we term as lane tokens. The lane tokens are then processed through the attention-based mechanism, resulting in new lane tokens with correlation information. The new lane token features are used to replace the original lane features in the output feature map of the first stage network, resulting in a refined feature map. Finally, the refinement head produce new and more accurate lane detection proposals through another row-wise existence and probability predictions from the refined feature map.

B. Feature Extractor

The feature extractor is responsible for encoding the point cloud raw data into an output feature map that is used by the detection head to make the final predictions. It is composed of two parts: the BEV encoder and the global feature correlator (GFC) backbone. Given a point cloud $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_n\}$, where $\mathbf{p}_i \in \mathbb{R}^{(3+C)}$ is a point in the 3D space with C additional features such as intensity and reflectivity, the feature extractor first encode the raw point cloud data into a pseudo BEV image of size $C_{BEV} \times H_{BEV} \times W_{BEV}$, where C_{BEV} is the number of feature channels, H_{BEV} is the number of rows, and W_{BEV} is the number of columns.

After obtaining the pseudo BEV image, the backbone then learn important features through global feature correlator. This results in the final output feature map of size $C_{head} \times H_{BEV} \times W_{BEV}$. We utilize the same feature extractor as seen in previous state-of-the-art network [9] as our focus in this work is on the detection head with the two-stage row-wise formulation.

C. Row-wise Detection Head

The row-wise detection head uses the final output feature map as an input and produce two predictions: the row-wise lane existence and the row-wise lane location probability. To do so, we leverage the fact that lane lines from a LiDAR scan have almost no shape distortion along the BEV map rows, thus, it is suitable to utilize shared-MLPs. As shown in Fig. 2, the MLPs are shared along the rows of the feature map, that is, each row in the feature map is considered as an individual feature vector (colorized as purple in Fig. 2) to be processed by the same MLPs.

We use two distinct shared-MLPs, one for predicting whether a lane line class exists on the road (row-wise lane existence), the other for predicting the horizontal (column-wise) location on which the lane line resides on (row-wise location probability) as shown in Fig. 2. Specifically, the row-wise existence MLPs output is an $N_{cls} \times H_{BEV} \times 2$ logits feature map, where N_{cls} is the number of lane line classes, and 2 is the number of parameters for predicting the probability of existence (i.e., exist or not), with column 0 represent not-exist flag, while column 1 represent exist flag. The row-wise probability MLPs output is an $N_{cls} \times H_{BEV} \times W_{BEV}$ logits feature map. To obtain the row-wise existence and location probability, we apply softmax function along the column to the logits of feature maps. The existence and the location of lane line of class c on row h is determined through the argmax function along the rows. Note that while each rows

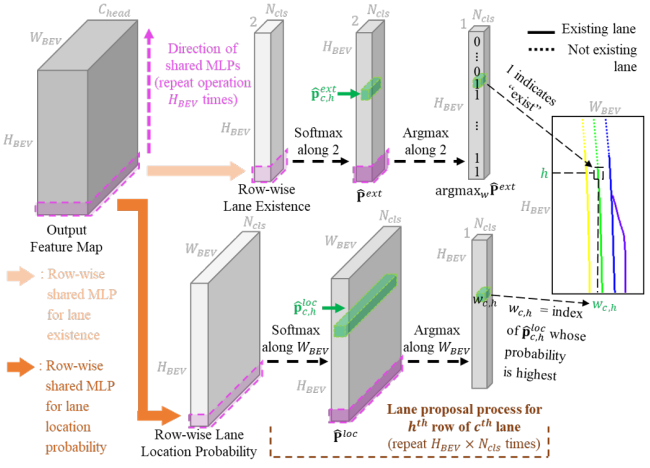

Fig. 2. Detailed process of row-wise detection head.

contains lane location probability, only rows with a positive existence prediction are considered as the first-stage lane proposals.

D. Refinement with Lane Correlation

After obtaining the first lane proposals from the detection head, the network further improve its predictions through the refinement head as shown in Fig. 1 (D). The refinement head first collects features from the final output feature map based on the location of lane proposals, resulting in a set of lane tokens. However, a lane class that does not have enough information may adversely affect the correlation operation, so we only extract features of lane classes that have more than a certain amount of positive existence along the entire rows.

To be specific, suppose that the lane existence predictions are $\hat{\mathbf{P}}^{ext} \in \mathbb{R}^{N_{cls} \times H_{BEV} \times 2}$ as described in the previous subsection. We collect features of lane class c iff.,

$$\frac{1}{H_{BEV}} \sum_{h}^{H_{BEV}} argmax_w(\hat{\mathbf{p}}^{ext}_{c,h}) > T_{ext}, \tag{1}$$

i.e., we only consider the lane classes that exists on more than $T_{ext} \cdot H_{BEV}$ number of rows.

The lane tokens have a size of $N_{lanes} \times H_{BEV} \times (C_{head} \times W_{thick})$, where N_{lanes} is the number of extracted lanes following the requirement stated in equation 1, and W_{thick} is the thickness of the lane that we extract when collecting the tokens. For an example, the thickness of three will result in a token that is constructed from the feature vector at the same coordinate as the lane proposal, and two other feature vectors from the neighboring columns.

The collected lane tokens are then processed by a transformer encoder block [17] to produce refined feature vectors. Intuitively, the transformer encoder block tries to find important features in the lane tokens while considering the interaction between feature vectors of different lanes via correlation (i.e., dot product of query and key). Such mechanism may be advantageous considering that most lane lines are constructed with a certain degree of regularity in terms of shape and distance between each other.

The lane tokens are then returned back to their original coordinates on the output feature map to create a refined feature map. This refined feature map is then used to predict the final row-wise existence and row-wise probability through two distinct shared-MLPs as in the first stage prediction. As such, the lanes, which are difficult to be detected in the first proposal, can be detected through interaction with other lane lines. This hypothesis is backed by both quantitative and qualitative enhancement, especially for hard cases such as severe occlusions.

E. Loss Function

The proposed row-wise detection network is supervised through two loss functions: lane existence loss $\mathscr{L}_{ext}$ and lane location probability loss $\mathscr{L}_{loc}$. As we mentioned earlier, since the row-wise detection method is formulated as predicting the row-wise probability, both losses are constructed as sum of row-wise cross-entropy loss. To be specific, the $\mathscr{L}_{ext}$ penalizes the network when it miss-classifies the existence of a certain lane class on the scene. Let $\mathbf{P}^{ext} \in \mathbb{R}^{N_{cls} \times H_{BEV} \times 2}$ be the lane existence ground truth where N_{cls} is the number of lane classes, H_{BEV} is the number of rows, and $\mathbf{p}^{ext}_{c,h} \in \mathbb{R}^2$ is a one-hot-encoded vector that indicates the existence of a lane of class c on row h. That is, $p^{ext}_{c,h,1} = 1$ if there exists a lane line of class c on row h, and $p^{ext}_{c,h,0} = 1$ if the opposite is true. Given the lane existence predictions $\hat{\mathbf{P}}^{ext} \in \mathbb{R}^{N_{cls} \times H_{BEV} \times 2}$, we define the lane existence loss as,

$$\mathscr{L}_{ext} = -\frac{1}{N_{cls} \cdot H_{BEV}} \sum_{c}^{N_{cls}} \sum_{h}^{H_{BEV}} \sum_{w}^{2} p^{ext}_{c,h,w} \cdot log(\hat{p}^{ext}_{c,h,i}). \tag{2}$$

For the lane location loss, suppose that $\mathbf{P}^{loc} \in \mathbb{R}^{N_{cls} \times H_{BEV} \times W_{BEV}}$ is the lane location probability ground truth, where W_{BEV} is the number of columns. $\mathbf{p}^{loc}_{c,h}$ is a one-hot encoded vector that indicates the location of a lane of class c on row h, that is, $p^{loc}_{c,h,w} = 1$ if there exists a lane line of class c on row h and column w, and $p^{loc}_{c,h,w} = 0$ if the opposite is true. Given the lane location probability $\hat{\mathbf{P}}^{loc} \in \mathbb{R}^{N_{cls} \times H_{BEV} \times W_{BEV}}$, we define the lane location probability loss as,

$$\mathscr{L}_{loc} = -\frac{1}{\sum_{c}^{N_{cls}} \sum_{h}^{H_{BEV}} p^{ext}_{c,h,1}} \sum_{c}^{N_{cls}} \sum_{h}^{H_{BEV}} \sum_{w}^{W_{BEV}} p^{loc}_{c,h,w} \cdot log(\hat{p}^{loc}_{c,h,w}) \cdot p^{ext}_{c,h,1}. \tag{3}$$

Unlike existence loss, the location loss is applied only to lane location probabilities for which its corresponding existence prediction is a positive flag. This is because predicting the lane locations of non-existent lanes may have adverse effects to the training of the network.

Both existence and location loss are applied to both prediction results of the first and second stage heads. Since both existence and location loss are normalized to each row (i.e., divided with $N_{cls} \cdot H_{BEV}$ and $\sum_{c}^{N_{cls}} \sum_{h}^{H_{BEV}} p^{ext}_{c,h,1}$, respectively), the total loss function is the summation of both lane existence loss and lane location probability loss,

$$\mathscr{L}_{total} = (\mathscr{L}_{ext} + \mathscr{L}_{loc})_{1^{st} stage} + (\mathscr{L}_{ext} + \mathscr{L}_{loc})_{2^{nd} stage}. \tag{4}$$

IV. Experiments

In this section, we provide the implementation details that are required to reproduce our results. Then we show and discuss our experimental results on the K-Lane dataset [9].

A. Dataset & Metric

We utilize the K-Lane dataset [9] throughout the experiments. The dataset contains a collection of up to six labelled lane lines on over 15k frames of LiDAR point clouds, where the labels are provided in the form of BEV images. Additionally, K-Lane includes carefully calibrated front camera images, which simplifies the visualization of LiDAR lane detection network output results as shown in Fig. 3.

For a fair comparison, we validate the performance with the F1-score, a harmonic mean of the precision and recall, following the previous work. However, we measure the

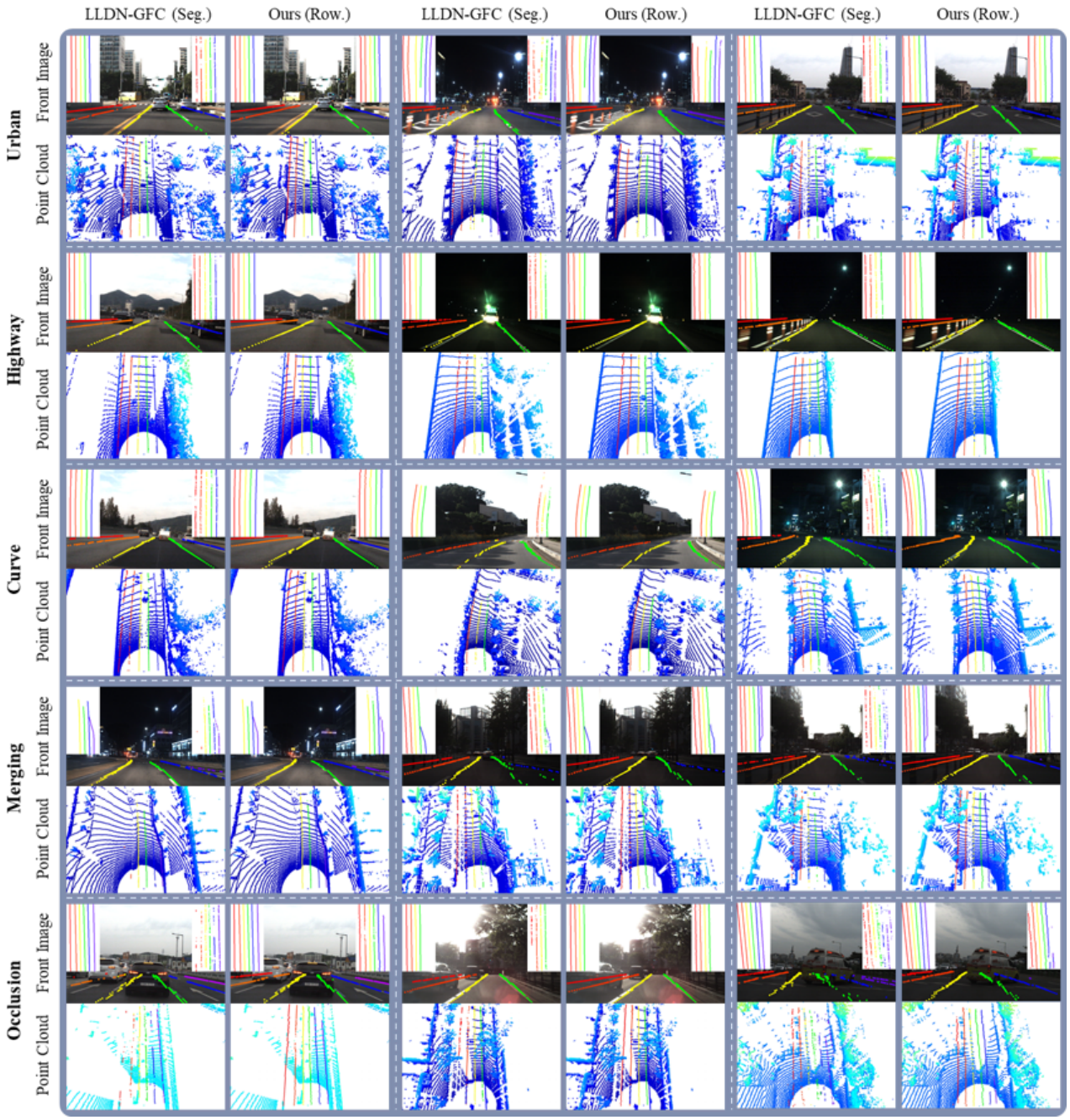

Fig. 3. Qualitative comparison of segmentation-based (SOTA prior work) and row-wise-based (ours): We show lane detection results in a total of five different conditions, i.e. urban, highway, curve, merging, and occlusion. We project the lane proposals onto the RGB camera image, where the BEV label is shown on the top-left corner, and the BEV prediction is shown on the top-right corner. We also visualize the lane proposals on the point cloud under its corresponding RGB camera image.

TABLE I

COMPARISON OF F1-SCORE ON VARIOUS ROAD CONDITIONS & GFLOPS

Model	Overall	Daylight	Night	Urban	Highway	Curve	Merging	Num. of Occluded Lines					GFLOPs
								0	1	2	3	4-6	
LLDN-GFC [9]	82.12	82.22	82.00	**81.75**	82.55	**78.05**	**81.08**	82.97	81.43	81.28	78.67	75.92	558.0
Ours (1-stage)	82.37	82.04	82.75	81.14	83.85	76.49	79.78	83.33	81.52	81.57	77.91	76.36	**385.1**
Ours (2-stage)	**82.74**	**82.58**	**82.92**	81.64	**84.05**	76.16	79.92	**83.44**	**81.87**	**82.00**	**80.37**	**79.16**	387.5

absolute computation of the network through floating point operations (FLOPs) rather than the inference speed as the speed measurements can vary depending on the quality of the hardware. Thus, in the experiments, we assess precision, recall, and computational complexity, all of which are crucial for lane detection task.

B. Implementation Details

We implement our experiments using PyTorch [18] on an Ubuntu 18.04 machine with RTX3090 GPUs. Unless stated otherwise, we set the maximum epochs to be 20 with a batch size of 4. We use Adam [19] as the optimizer, with a learning rate of 10^{-4}. All of the experiments are conducted on the K-Lane dataset [9].

C. Results on the K-Lane Dataset

Fig. 3 and Table I respectively show the qualitative and the quantitative results of our proposed network compared to the existing baseline, LiDAR lane detection network with global feature correlator (LLDN-GFC) based on segmentation approach [9]. The row-wise detection network outperforms the existing segmentation-based network in terms of F1-score by +0.62 points. Moreover, the row-wise detection network achieves state-of-the-art performance with less computational complexity, where the GFLOPs is about 30% less when being compared to the existing baseline. Though it is obvious that the overall performance of the model can be dependent on the hyper-parameter values as shown in Table II, we note that the proposed network outperforms segmentation-based for most road conditions. In particular, when more than four lanes are occluded, the proposed network shows a significant improvement of 3.24% over the existing SOTA neural network. To the best of our knowledge, there is no publicly available LiDAR lane detection network other than the LLDN-GFC at the time of submission.

The superior performances of the proposed two-stage LiDAR lane detection network with row-wise approach may originate from three aspects. First, the row-wise approach allows the detection head to directly produce the prediction values for all columns in the same row through the highly-optimized MLP operation. This is in contrast with the segmentation-based approach, where the prediction values are obtained by applying 1×1 convolution to each column, which may result in a more inefficient operation. Second, by applying the MLP per row, the detection head of the network may directly learn the correlation between all features on the same row in the feature map. On the contrary, the each of the segmentation-based predictions is produced only from that specific grid in the feature map. Third, the second-stage refinement through lane correlation enables the network to learn important connection between lane features globally, thus helping in the case of severe lane occlusions. As shown in Fig. 4, the second-stage proposals, which reflects the correlation of the first-stage proposals as lane tokens, robustly detect the lanes (colorized in purple) which were not detected in the 1st-stage proposals due to vehicle occlusions.

Quantitatively, as shown in Table I, for the single-stage neural network, we observe only a 0.44% improvement over prior SOTA network in the roads where more than 4 to 6 lanes are occluded. On the other hand, for the same occlusion level, the proposed two-stage network improves the SOTA by 3.24%.

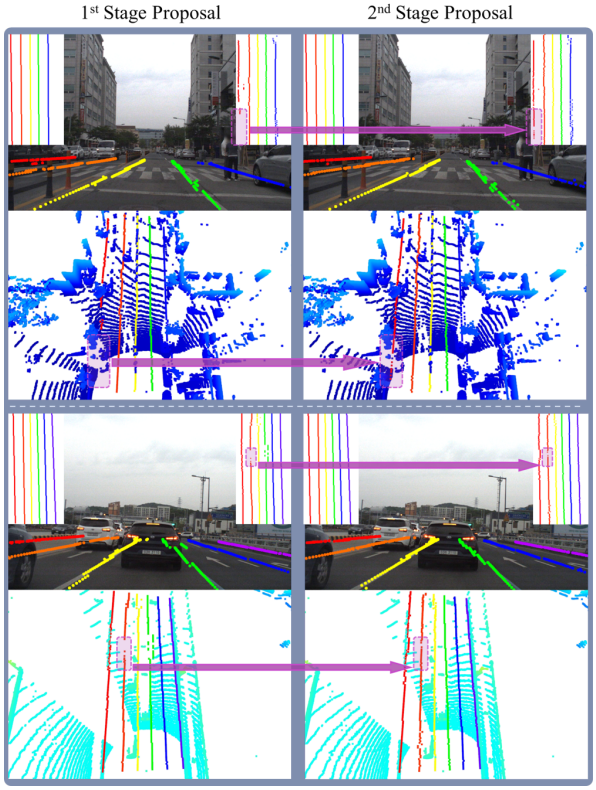

Fig. 4. Qualitative comparison of 1^{st}-stage and 2^{nd}-stage lane proposals from the proposed network. The 1^{st}-stage lane proposals is the intermediate output of the proposed network which is the input of 2^{nd}-stage detection head (i.e., Refinement head). We project the lane proposals onto the RGB camera image, where the BEV label is shown on the top-left corner, and the BEV prediction is shown on the top-right corner. We also visualize the lane proposals on the point cloud under its corresponding RGB camera image.

TABLE II
COMPARISON OF PERFORMANCE FOR VARIOUS HYPER-PARAMETERS

Backbone	2-Stage Head			Overall	GFLOPs
Depth	T_{ext}	Depth	W_{thick}	F1-score	
1	-	-	-	79.52	**379.64**
3	-	-	-	82.37	385.08
5	-	-	-	82.25	390.52
2	0.3	1	5	82.57	384.76
3	0.3	1	5	**82.74**	387.48
5	0.3	1	5	82.57	392.92
3	0.3	3	5	82.39	387.70
3	0.3	5	5	82.63	387.90
3	0.5	1	5	82.49	387.40
3	0.7	1	5	82.54	387.24
3	0.3	1	3	82.71	387.44
3	0.3	1	7	82.52	387.54

D. Experiments on Design Choices

There are several hyperparameter choices that affect the final performance of the proposed network. On the first-stage network, the depth of the transformer-based global feature correlator (GFC-T) [9], which partly reflect the model capacity, should be optimal w.r.t. the complexity of the problem and the number of samples in the dataset. On the second-stage network, there are three hyperparameters: the lane existence threshold (T_{ext}), network depth, and the thickness of extracted lanes when collecting the lane tokens (W_{thick}).

To study the effect of each hyperparameter, we conduct several experiments with different hyperparameter values. As seen in Table II, all previously mentioned hyperparameters are affecting the network performance.

From the experimental results, we find that setting the backbone depth to be three yields the best performance in terms of F1-score. Interestingly, the optimal depth for the second stage refinement head is found to be one, where deeper network lead to lower F1-score. In addition, we observe that setting W_{thick} to be five gives the optimal performance compared to other values.

We also perform ablation study on the effect of the second-stage network. From Table I, we can see that the proposed two-stage row-wise detection network is especially robust on the occlusion cases, which empirically support our explanation on Section 3.

V. Conclusions

In this work, we have proposed a novel two-stage LiDAR lane detection network with row-wise detection approach. The first stage network is responsible for predicting lane proposals, while the second stage network refines the first stage feature map through attention-based mechanism between lane tokens, before predicting the refined lane proposals. From experimental results on the K-Lane dataset, the proposed network advances the state-of-the-art performance with an overall F1-score of 82.74% by 0.62% while reducing the GFLOPs by about 30%. In particular, the proposed network greatly improves the lane detection performance for severe occlusion cases. Therefore, the proposed network enables a robust lane detection required for safe autonomous driving in the congested traffics where performance deterioration is previously significant.

ACKNOWLEDGMENT

This work was partly supported by National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021R1A2C3008370).

References

[1] D. Schreiber, B. Alefs, and M. Clabian, "Single camera lane detection and tracking," in *Proceedings. 2005 IEEE Intelligent Transportation Systems, 2005.* IEEE, 2005, pp. 302–307.

[2] Y. Dong, J. Xiong, L. Li, and J. Yang, "Robust lane detection and tracking for lane departure warning," in *2012 International Conference on Computational Problem-Solving (ICCP).* IEEE, 2012, pp. 461–464.

[3] G. Deng and Y. Wu, "Double lane line edge detection method based on constraint conditions hough transform," in *2018 17th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES).* IEEE, 2018, pp. 107–110.

[4] Z. Qin, H. Wang, and X. Li, "Ultra fast structure-aware deep lane detection," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIV 16*. Springer, 2020, pp. 276–291.

[5] H. Abualsaud, S. Liu, D. Lu, K. Situ, A. Rangesh, and M. M. Trivedi, "Laneaf: Robust multi-lane detection with affinity fields," 2021.

[6] L. Liu, X. Chen, S. Zhu, and P. Tan, "Condlanenet: a top-to-down lane detection framework based on conditional convolution," 2021.

[7] 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.

[8] TuSimple, "Tusimple benchmark," https://github.com/TuSimple/tusimple-benchmark, 2017, accessed: 2021-09-08.

[9] D.-H. Paek, S.-H. Kong, and K. T. Wijaya, "K-lane: Lidar lane dataset and benchmark for urban roads and highways," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2022.

[10] L. Tabelini, R. Berriel, T. M. Paixao, C. Badue, A. F. De Souza, and T. Oliveira-Santos, "Keep your eyes on the lane: Real-time attention-guided lane detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 294–302.

[11] J. Philion, "Fastdraw: Addressing the long tail of lane detection by adapting a sequential prediction network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 582–11 591.

[12] T. Ogawa and K. Takagi, "Lane recognition using on-vehicle lidar," in *2006 IEEE Intelligent Vehicles Symposium*. IEEE, 2006, pp. 540–545.

[13] P. Lindner, E. Richter, G. Wanielik, K. Takagi, and A. Isogai, "Multi-channel lidar processing for lane detection and estimation," in *2009 12th International IEEE Conference on Intelligent Transportation Systems*. IEEE, 2009, pp. 1–6.

[14] D. C. Hernandez, V.-D. Hoang, and K.-H. Jo, "Lane surface identification based on reflectance using laser range finder," in *2014 IEEE/SICE International Symposium on System Integration*. IEEE, 2014, pp. 621–625.

[15] M. Bai, G. Mattyus, N. Homayounfar, S. Wang, S. K. Lakshmikanth, and R. Urtasun, "Deep multi-sensor lane detection," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3102–3109.

[16] P. Martinek, G. Pucea, Q. Rao, and U. Sivalingam, "Lidar-based deep neural network for reference lane generation," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 89–94.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[18] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, pp. 8026–8037, 2019.

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.