Supplemental Materials

Recursive Video Lane Detection

Dongkwon Jin
Korea University
dongkwonjin@mcl.korea.ac.kr

Dahyun Kim
Korea University
dhkim@mcl.korea.ac.kr

Chang-Su Kim
Korea University
changsukim@korea.ac.kr

A. Implementation Details

A.1. ILD architecture

ILD consists of a single encoder and two decoders in series. Figure 1 shows the structure of ILD.

Encoding: Given an image I, the encoder extracts a convolutional feature map $X \in \mathbb{R}^{H \times W \times K}$. First, we extract multi-scale feature maps using ResNet18 [1] as the backbone. Then, we combine the three coarsest maps. Specifically, we match the channel dimensions of the feature maps to K. We then match the resolutions of the two coarser maps to the finest one via bilinear interpolation and concatenate them. From the concatenated feature map, we obtain X via convolutional and up-sampling layers.

Decoding: From the feature map X, the two decoders sequentially generate a probability map $P \in \mathbb{R}^{H \times W \times 1}$ and a coefficient map $C \in \mathbb{R}^{H \times W \times M}$. First, we obtain the probability map P using a series of convolutional layers and a sigmoid function. Then, we estimate the coefficient map C from P. To this end, we increase the channel dimension of P to K and combine the result with a sinusoidal positional bias $B \in \mathbb{R}^{H \times W \times K}$ [2] via element-wise addition. From the combined feature map, we generate an offset map $O \in \mathbb{R}^{H \times W \times 18}$ and a transformed feature map and then apply a deformable convolution [3] with a 3×3 kernel to regress C.

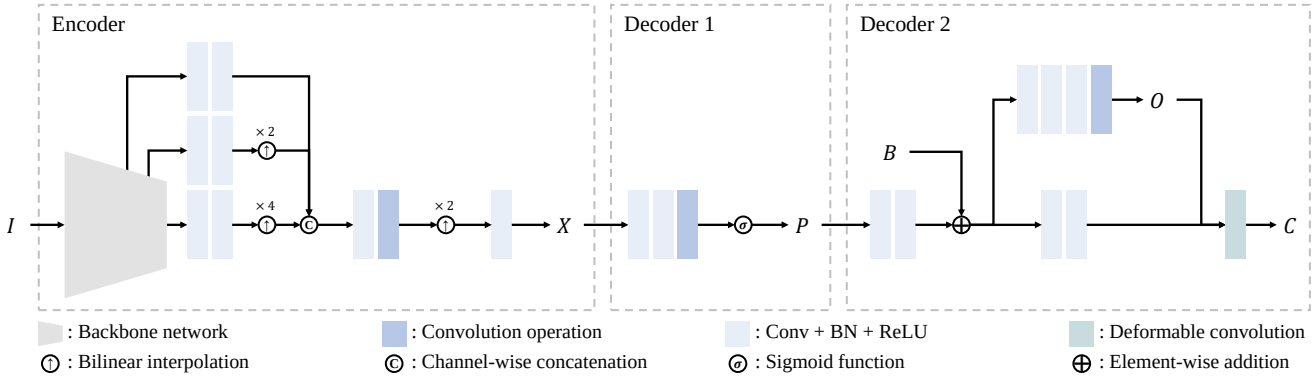

Figure 1. The structure of the proposed ILD.

A.2. PLD architecture

PLD consists of motion estimation and feature refinement modules in Figure 2.

Motion estimation: We first apply a pair of convolution layers to feature maps $\tilde{X}^t$ and X^{t-1}, respectively. Then, we construct a cost volume $V \in \mathbb{R}^{H \times W \times D^2}$ by first computing local correlations and then performing the softmax operation. We concatenate the cost volume V with $\tilde{X}^t$ and then apply convolutions and down-sampling operations to obtain the downsampled motion field $F_{\text{down}} \in \mathbb{R}^{H/4 \times W/4 \times 2}$.

Feature refinement: From F_{down}, we obtain the upsampled motion field $F \in \mathbb{R}^{H \times W \times 2}$ via bilinear interpolation. We then backward warp the previous output X^{t-1} and L^{t-1} and obtain $X^{t-1}_{\text{warp}} \in \mathbb{R}^{H \times W \times K}$ and $L^{t-1}_{\text{warp}} \in \mathbb{R}^{H \times W \times 1}$. Then, from the

warped lane mask L_{warp}^{t-1}, we obtain a guidance feature map $G^{t-1} \in \mathbb{R}^{H \times W \times K}$ by applying 2D convolution layers. Lastly, we concatenate the three feature maps $\tilde{X}^t$, X_{warp}^{t-1}, and G^{t-1} in the channel dimension and aggregate them to produce the refined feature map $X^t \in \mathbb{R}^{H \times W \times K}$ through 2D convolution layers.

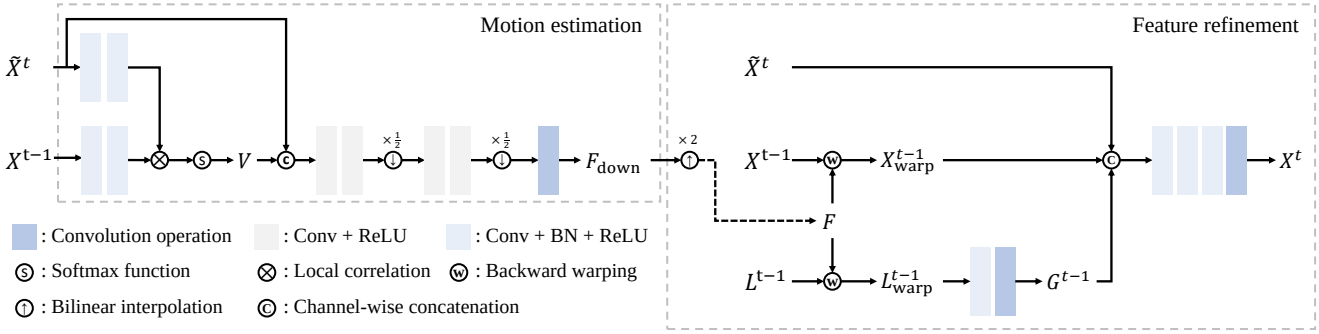

Figure 2. The structure of the proposed PLD.

A.3. Training and hyper-parameters

To train ILD and PLD, we generated a ground-truth (GT) lane probability map $\bar{P} \in \mathbb{R}^{H \times W \times 1}$ and a GT coefficient map $\bar{C} \in \mathbb{R}^{H \times W \times M}$. For each image, $\bar{P}$ was straightforwardly obtained from the lane mask: $\bar{P}_x = 1$ if pixel x belongs to a lane, and $\bar{P}_x = 0$ otherwise. To obtain $\bar{C}$, we first extracted M eigenlanes using all lanes in a training set, as done in [4]. Then, we transformed each lane in an image to an M-dimensional coefficient vector. In the image, $\bar{C}_x$ was assigned the coefficient vector, if x belongs to the lane. Otherwise, if x does not belong to any lane, $\bar{C}_x$ was assigned the zero vector. In such a case, the loss $\ell_{\text{reg}}(C_x, \bar{C}_x)$ in (7) in the main paper was not computed during the training.

We used the AdamW optimizer [5] with an initial learning rate of 10^{-4} and halved it after every 80,000 iterations five times. Also, we used a batch size of eight for 400,000 iterations and augmented training images by randomly flipping them horizontally. For both VIL-100 and OpenLane-V datasets, we resized training images to 384×640.

Table 1 lists the hyper-parameters in the proposed RVLD algorithm. We will make the source codes publicly available.

Table 1. Hyper-parameter setting.

Hyper-parameters	Descriptions
$H = 96$	Feature height
$W = 160$	Feature width
$K = 64$	# of feature channels
$M = 6$	# of eigenlanes
$D = 5$	Search window size in Figure 6 in the main paper
$N = 330$	# of samples in each lane
$R = 3$	Rank of the factored matrix in ALS in (9)

B. OpenLane-V

As described in Section 4.1 in the main paper, only visible lane parts were annotated in the OpenLane dataset [6], so we improved the annotations in OpenLane and constructed a modified dataset, called OpenLane-V, by filling in missing parts semi-automatically based on matrix completion [7]. This semi-automatic process consists of four steps: preprocessing, matrix factorization, manual annotation, and postprocessing. Let us describe each step in detail.

Preprocessing: In OpenLane, the same lane may be split into multiple parts and some of them are too short, as shown in the middle row in Figure 3. Such short lane parts make the matrix completion task unstable since they increase the sparsity of a matrix. Thus, we performed preprocessing to connect lane parts originating from the same lane. To link them automatically, we define the distance between a pair (l_1, l_2) of lane parts as follows. First, if there is an overlap between vertical coordinates of l_1 and l_2, the distance $d(l_1, l_2)$ is set to infinity because they cannot come from the same lane. Suppose that there is no overlap, l_1 is the upper part, and l_2 is the lower part. Then, we define $d(l_1, l_2)$ as the angle between the tangent line to the bottommost point in l_1 and the tangent line to the topmost point in l_2. Using the distances of all pairs of lane parts, we performed greedy matching: We connected the pair of lane parts with the shortest distance and removed them from the set

of merging candidates. We repeated this process until there was no matching pair with a distance less than a threshold. The bottom row in Figure 3 shows some examples of connected lanes.

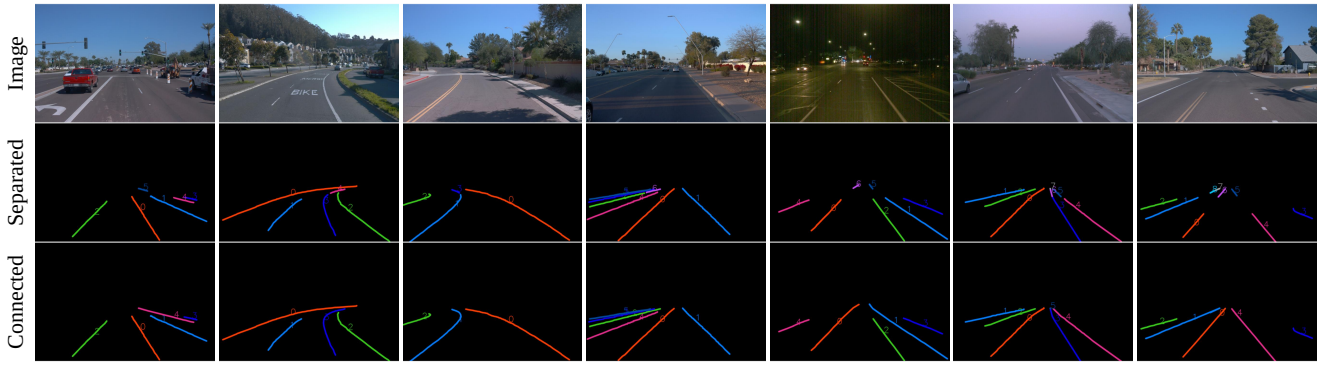

Figure 3. In the middle row, some lanes are separated but they actually belong to the same lane. In the bottom row, such lanes are linked through the preprocessing.

Matrix factorization: To fill in missing parts semi-automatically based on matrix completion [7], we obtained a lane matrix $\mathbf{A} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_L] \in \mathbb{R}^{N \times L}$ containing L lanes in the dataset, where each lane $\mathbf{x}_i = [x_1, x_2, \ldots, x_N]^\top$ was represented by the x-coordinates of N points sampled uniformly in the vertical direction. Here, some lanes were already merged through the preprocessing step. Then, we performed factorization $\mathbf{A} \approx \mathbf{U}^\top \mathbf{V}$, where $\mathbf{U} \in \mathbb{R}^{R \times N}$ and $\mathbf{V} \in \mathbb{R}^{R \times L}$. To find optimal $\mathbf{U}$ and $\mathbf{V}$, we adopted the alternating least square (ALS) process [8, 9], in which we optimized $\mathbf{V}$ after fixing $\mathbf{U}$, and vice versa. We set $N = 330$ and $R = 3$. Also, we set the regularization parameter $\lambda = 1 \times 10^{-12}$ in (9) in the main paper. We implemented the ALS process using the PySpark library [10].

Manual annotation: However, entirely invisible lanes were not annotated in OpenLane. It is impossible to restore such invisible lanes automatically. They disappear suddenly in videos and cause flickering artifacts, as illustrated in Figure 4. Also, we experimentally discovered that too short lanes might be inpainted unreliably. To cope with these issues and improve the temporal consistency of lanes, we manually annotated some lanes using the information in neighboring frames. To annotate new lanes or refine unreliable results, we employed an annotation tool, CVAT [11], as shown in Figure 5. The manual annotation process took more than 100 man-hours.

Figure 4. In OpenLane, totally invisible lanes are not annotated. They cause flickering artifacts as depicted by dotted red lines. To overcome this issue, we manually annotate such lanes using the information in adjacent frames.

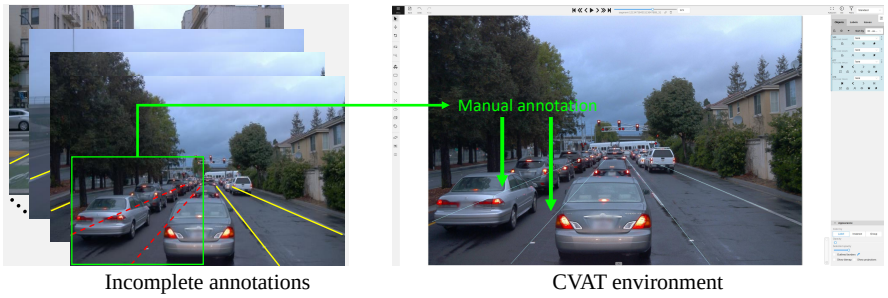

Figure 5. Manual annotation process of lanes using CVAT. Missing annotations are depicted by dotted red lines.

Postprocessing: We removed overlapping lanes representing the same lane through non-maximum suppression. Then, we considered only ego and alternative lanes and filtered out the other lanes to focus on the important lanes for driving. We also performed temporal smoothing of lanes in each frame using six neighboring frames. Specifically, for each lane, we found the matching lane instances in three past and three future frames, respectively. Then, we adjusted the target lane by applying a Gaussian smoothing filter to the positions of the target and matching lanes.

We also removed some videos, in which completed lanes were unreliable in large portions of frames. Consequently, we processed 90K images in total and constructed OpenLane-V, which improved the annotations in OpenLane meaningfully. It will be made available publicly. Figure 6 shows some examples of improved annotations in OpenLane-V. It is recommended to watch the accompanying video for more examples.

Figure 6. In OpenLane, only visible lane parts are annotated, so the same lane may be split into multiple parts. In contrast, in OpenLane-V, each lane is seamlessly annotated, depicted by a yellow line, and invisible parts are inpainted.

C. Ablation studies

Table 2 compares the $\mathrm{F1}^{0.5}$, $\mathrm{R}_{\mathrm{F}}^{0.5}$, and $\mathrm{R}_{\mathrm{M}}^{0.5}$ scores of several ablated methods on OpenLane-V according to the number of past frames used. It also lists the processing times in seconds per frame (spf). Method I is the proposed RVLD using a single previous frame only, while Methods II and III use two and three past frames, respectively. Specifically, in Method II. we estimate two motion fields from I^t to I^{t-2} and I^{t-1}. Then, we warp the past information X^{t-2}, L^{t-2}, X^{t-1}, and L^{t-1} to the current frame I^t by employing the corresponding fields. After producing G^{t-2} and G^{t-1}, we obtain the refined feature map X^t by aggregating the warped information with $\tilde{X}^t$, similarly to (6) in the main paper. Method III performs a similar process using the information in I^{t-1}, I^{t-2}, and I^{t-3}. Methods IV, V, and VI are modified versions of I, II, and III, in which the refined features are not used in the future frames. Note that, below, Methods I and IV are the same as Methods V and IV in Table 4 in the main paper, respectively.

Compared to Method I, the performances of II and III fluctuate — but only slightly — according to the number of past frames used. In Methods IV, V, VI, similar tendencies are observed. Meanwhile, the processing speed gets slower with more past frames, since the motion estimation should be performed as many times and the feature refinement becomes more complicated. Therefore, as a tradeoff between the performance and the complexity, the proposed RVLD (Method I) exploits the information of a single previous frame only but does so recursively with the feature reuse scheme. As a result, RVLD exploits temporal correlation in a video both efficiently and effectively.

Table 2. Ablation studies of the proposed RVLD on OpenLane-V according to the number of past frames used. The processing times are reported in seconds per frame (spf).

	$\mathrm{F1}^{0.5}$	$\mathrm{R}_{\mathrm{F}}^{0.5}$	$\mathrm{R}_{\mathrm{M}}^{0.5}$	spf
I. RVLD	0.825	0.014	0.167	0.0125s
II. ILD + PLD (using 2 past frames)	0.829	0.015	0.170	0.0154s
III. ILD + PLD (using 3 past frames)	0.813	0.015	0.187	0.0179s
IV. ILD + PLD (w/o feature reuse)	0.822	0.017	0.172	0.0125s
V. ILD + PLD (using 2 past frames w/o feature reuse)	0.824	0.017	0.165	0.0154s
VI. ILD + PLD (using 3 past frames w/o feature reuse)	0.816	0.017	0.173	0.0179s

D. Experimental Results

D.1. Comparison on VIL-100

Figure 7 compares the proposed RVLD with conventional algorithms on VIL-100. RVLD provides better detection results than the conventional algorithms.

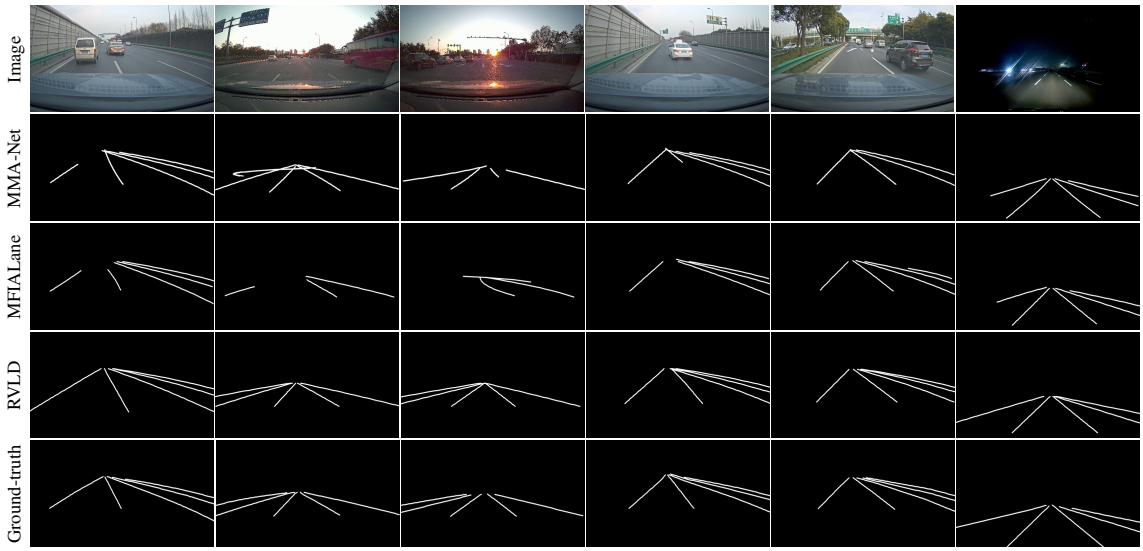

Figure 7. Comparison of lane detection results on the VIL-100 dataset.

Figure 8 compares the lane detection results of RVLD with those of the conventional algorithms for three consecutive video frames in VIL-100. We see that RVLD provides temporally more stable detection results than the conventional algorithms.

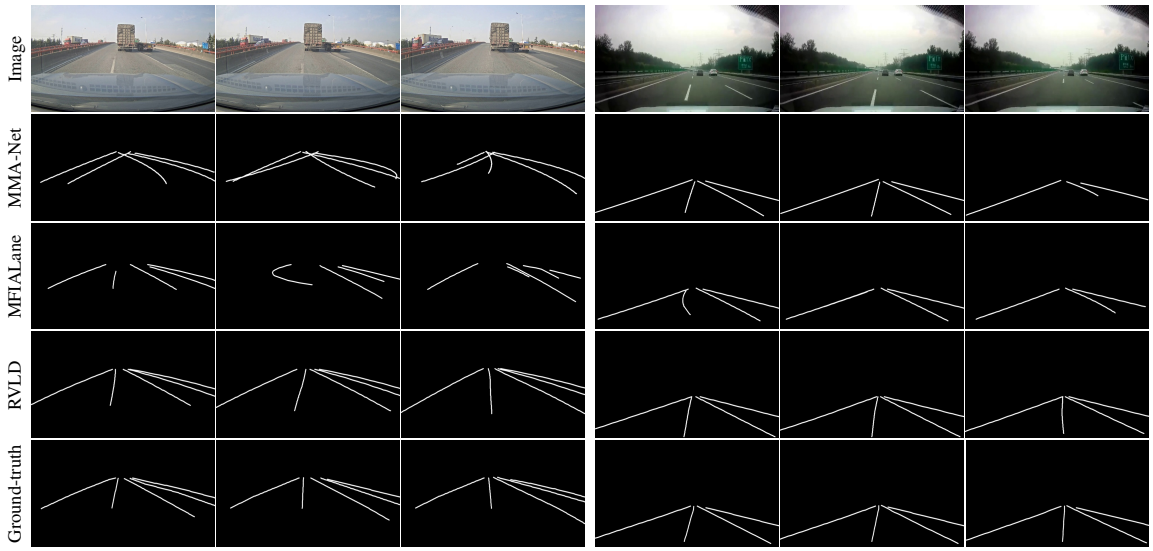

Figure 8. Comparison of lane detection results for three consecutive video frames in the VIL-100 dataset.

D.2. Comparison on OpenLane-V

Figure 9 compares the proposed RVLD with state-of-the-art image-based lane detectors on OpenLane-V. It is strongly recommended to watch the accompanying video to compare the temporal consistency of detected lanes.

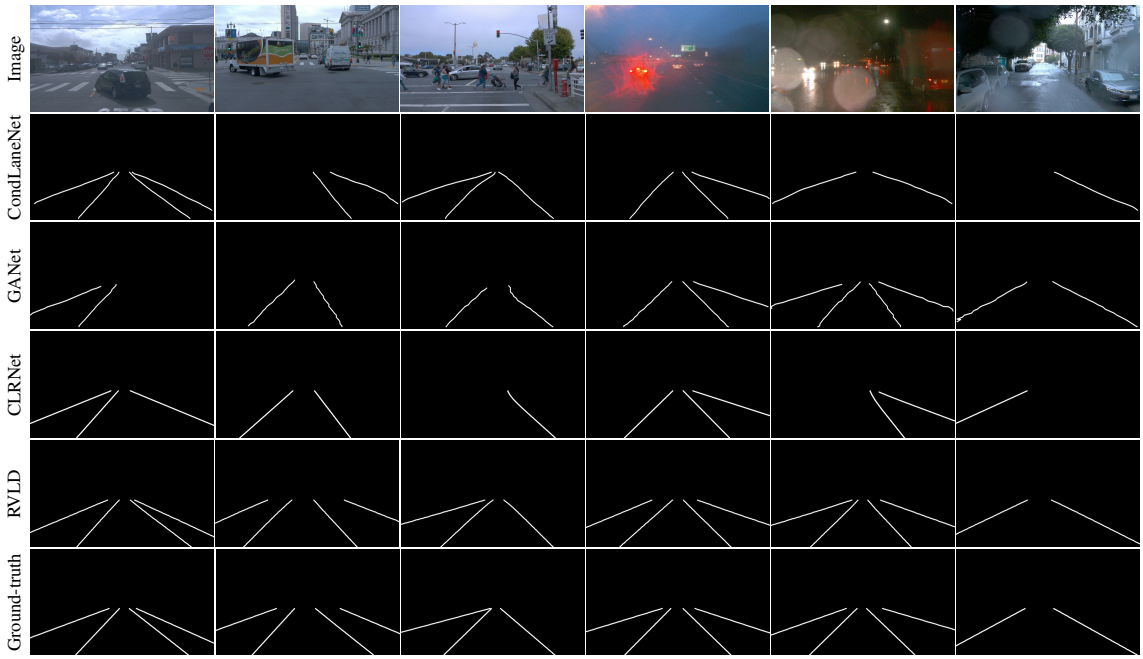

Figure 9. Comparison of lane detection results on the OpenLane-V dataset.

Figure 10 compares the proposed RVLD with existing video lane detectors on OpenLane-V. The proposed RVLD yields better detection results than the existing detectors.

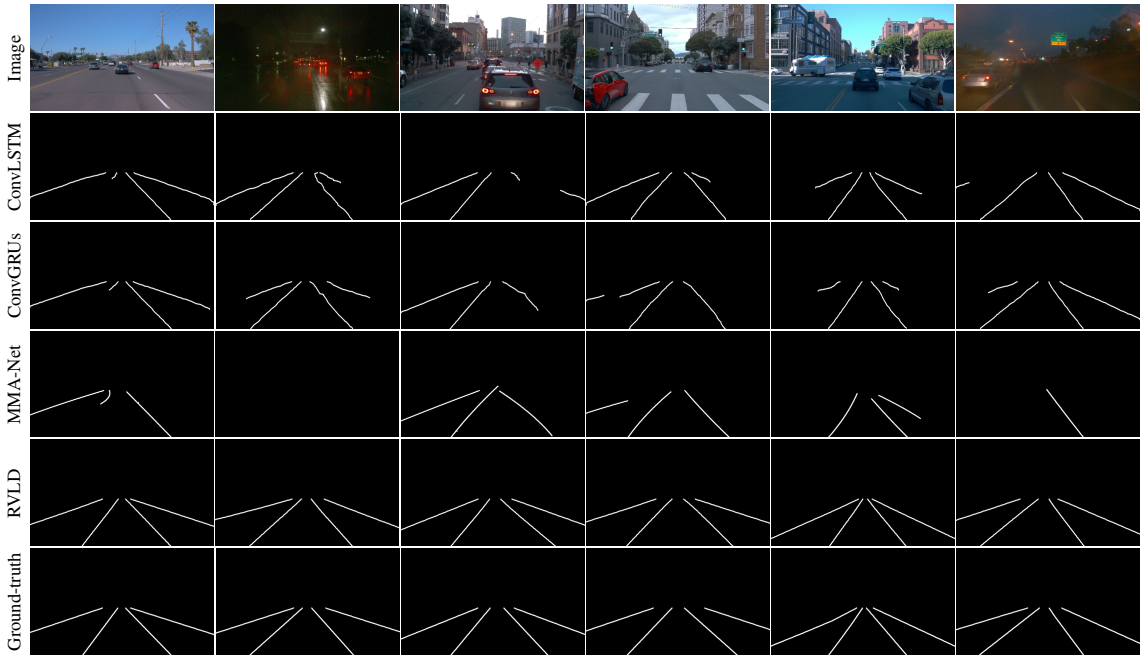

Figure 10. Comparison of lane detection results on the OpenLane-V dataset.

Figure 11 and Figure 12 compare the lane detection results of the proposed RVLD with those of the state-of-the-art image-based lane detectors for three consecutive video frames in OpenLane-V. Note that RVLD provides temporally more stable detection results.

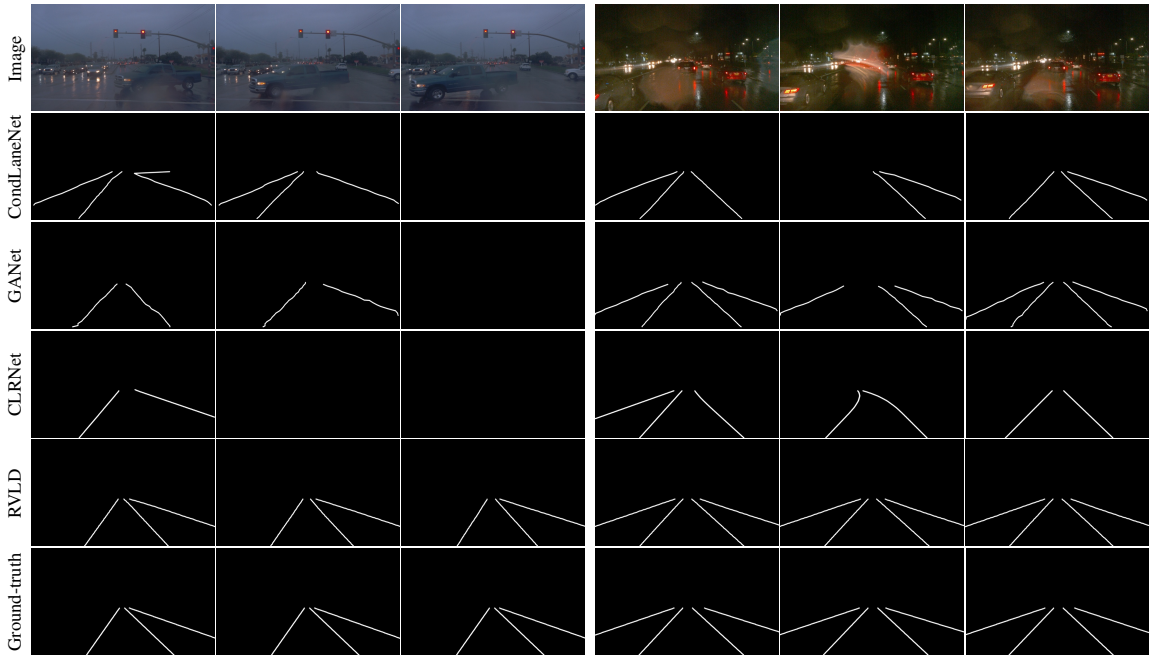

Figure 11. Comparison of lane detection results for three consecutive video frames in the OpenLane-V dataset.

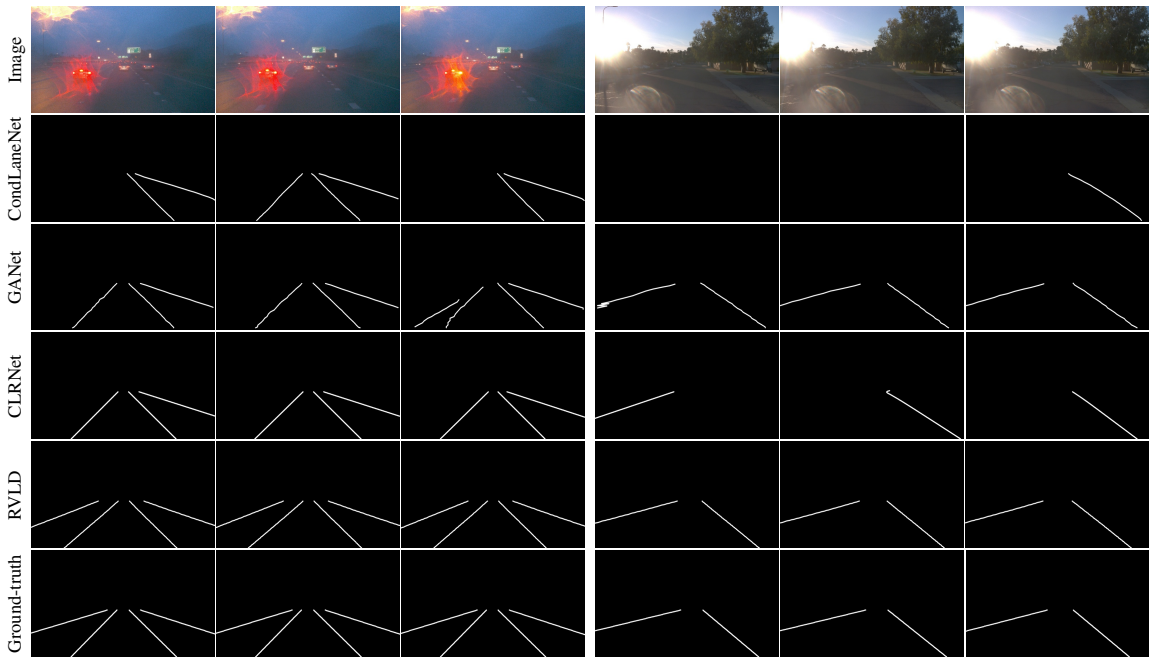

Figure 12. Comparison of lane detection results for three consecutive video frames in the OpenLane-V dataset.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE CVPR*, 2016. 1

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. NeurIPS*, 2017. 1

[3] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *Proc. IEEE ICCV*, 2017. 1

[4] D. Jin, W. Park, S.-G. Jeong, H. Kwon, and C.-S. Kim, "Eigenlanes: Data-driven lane descriptors for structurally diverse lanes," in *Proc. IEEE CVPR*, 2022. 2

[5] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proc. ICLR*, 2019. 2

[6] L. Chen, C. Sima, Y. Li, Z. Zheng, J. Xu, X. Geng, H. Li, C. He, J. Shi, Y. Qiao, and J. Yan, "PersFormer: 3D lane detection via perspective transformer and the OpenLane benchmark," in *Proc. ECCV*, 2022. 2

[7] E. J. Candès and T. Tao, "The power of convex relaxation: Near-optimal matrix completion," *IEEE Trans. Inf. Theory*, vol. 56, no. 5, pp. 2053–2080, 2010. 2, 3

[8] T. Hastie, R. Mazumder, J. D. Lee, and R. Zadeh, "Matrix completion and low-rank SVD via fast alternating least squares," *The Journal of Machine Learning Research*, vol. 16, no. 1, pp. 3367–3402, 2015. 3

[9] P. Jain, P. Netrapalli, and S. Sanghavi, "Low-rank matrix completion using alternating minimization," in *Proc. 45th Annual ACM Symp. Theory of Computing*, 2013. 3

[10] "Apache, PySpark." [Online]. Available: https://spark.apache.org/docs/latest/api/python/index.html. 3

[11] "CVAT." [Online]. Available: https://www.cvat.ai/. 3