LaneCPP: Continuous 3D Lane Detection using Physical Priors

Supplementary Material

# Surface Hypotheses	Pitch Angles
1	$\{0°\}$
3	$\{-2°, 0°, 2°\}$
5	$\{-2°, -1°, 0°, 1°, 2°\}$
15	$\{-5°, -2°, -1.7°, -1.3°, -1°, -0.7°, -0.3°, 0°, 0.3°, 0.7°, 1°, 1.3°, 1.7°, 2°, 5°\}$
27	$\{-10°, -8.5°, -7°, -5.8°, -4.5°, -3.3°, -2°, -1.7°, -1.4°, -1°, -0.8°, -0.6°, -0.3°, 0°, 0.3°, 0.6°, 0.8°, 1°, 1.4°, 1.7°, 2°, 3.3°, 4.5°, 5.8°, 7°, 8.5°, 10°\}$

Table 1. Different orientations of surface hypotheses.

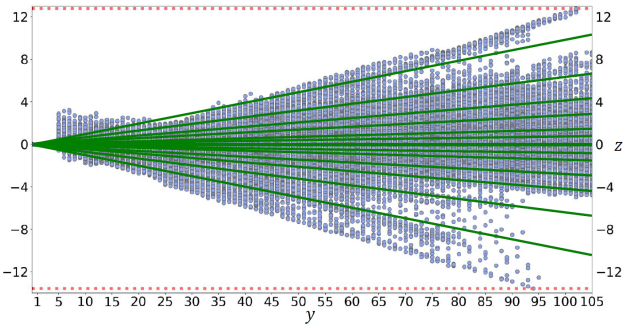


Figure 1. Height distribution (z) along the longitudinal direction (y) of ground truth line points (blue points) on OpenLane dataset. Height deviations in the near-range (left side) tend to be smaller than in the far-range (right side) spanning a triangle-like region of interest in the y-z-profile. For the spatial transformation, we sample surface hypotheses (green) of different pitch angles to cover this region.

1. Architecture Details

In the following section, we provide additional details regarding the model architecture.

1.1. Backbone

Similar to [2], we use a modified version of EfficientNet [13] as our backbone. More precisely, we extract a specific layer as the following module's input. Then, several convolution layers are applied, such that the backbone module outputs four different scaled front-view feature maps. Their resolutions are 180×240, 90×120, 45×60, 22×30. Each of the front-view feature maps is then fed into the spatial transformation module. The total number of parameters of the backbone is $10.28\,\mathrm{M}$.

1.2. Spatial transformation

The depth branch consists of two convolution layers each with 128 kernels and zero-padding, followed by batch norm and ReLU activation. An additional convolution layer uses S (number of surface hypotheses) kernels of size 1×1 followed by a channel-wise softmax to obtain the depth distribution. Since the depth distribution should be similar for all front-view feature maps of different scales, only one feature map needs to be propagated through the depth-branch. We use the feature map with lowest resolution 22×30 and repeat the resulting depth distribution of shape $22 \times 30 \times S$ (with S the number of surface hypotheses) at the neighboring feature cells to match the higher resolutions. Consequently, we obtain depth distributions for all scales of front-view feature maps sharing the same depth information.

To model the road surface's region of interest, we select surface hypotheses such that the distribution of lane height is covered (see Fig. 1). The surface hypotheses are planes crossing the origin of the 3D coordinate system with different orientations with respect to the pitch angle. The different configurations that we use in the experimental section are listed in Table 1.

After the front-view features are lifted to 3D space they are accumulated on BEV grids. Analogously to the multi-scale front-view feature maps, we also model multi-scale BEV feature maps. The different resolutions are 208×128, 104×64, 52×32, 26×16.

1.3. BEV feature fusion

The BEV feature fusion module consists of convolution layers operating on each scale to down-sample the higher resolutions to the lowest resolution feature map of shape 26×16. Afterwards, all feature maps are simply concatenated and fed through several layers preserving the resolution. Each contains a convolution with zero-padding, batch norm and ReLU activation. The last convolution layer uses 64 channels, thus, the input to the detection head is of shape $26 \times 16 \times 64$.

1.4. Detection head

The detection head operates on a BEV feature map of shape $26 \times 16 \times 64$ covering a range of $[-10\,\mathrm{m},\ 10\,\mathrm{m}]$ in lateral x-direction and $[3\,\mathrm{m},\ 103\,\mathrm{m}]$ in longitudinal y-direction. Based on the location of initial line proposals, features are pooled from the BEV feature map for each line proposal as illustrated in Fig. 2. More precisely, we step through a proposal inside the BEV feature grid with a small step size and determine the nearest cells, where the maximum number of cells is limited to max_cells. We then take the 64-dimensional features of the set of selected cells and flatten it to a feature vector of size $64 \cdot$ max_cells. If less than max_cells are pooled for the proposal, the remaining entries

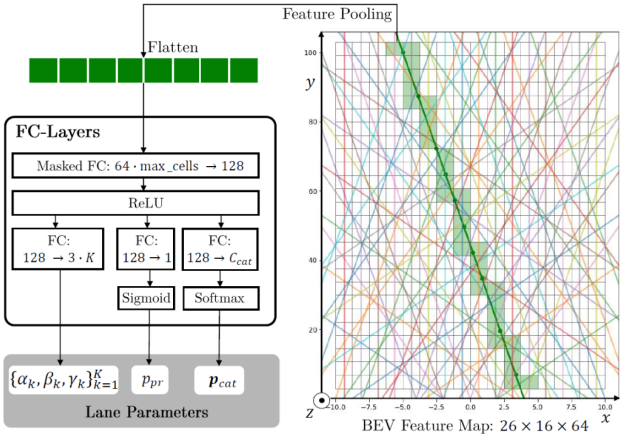


Figure 2. The detection head of our model: First, features are pooled from the BEV feature map for each proposal. Afterwards, pooled features are flattened and fed through several fully-connected (FC) layers, which share weights for all proposals, to finally obtain the lane parameters.

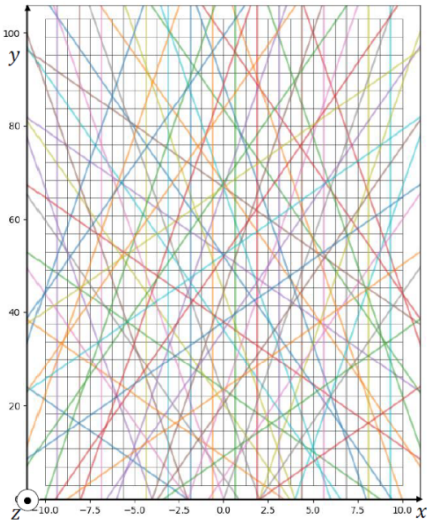


Figure 3. Visualization of different initial line proposals. Colorful lines represent the line proposals. The black lines show the grid of the final BEV feature map.

of the feature vector are simply masked out. The resulting feature vector for each line proposal is then propagated through the fully-connected layers as depicted in Fig. 2. Important to notice is also that the fully connected layers share weights among all proposals to learn the same patterns for different line orientations from the BEV feature map. Finally, for each proposal the model yields parameters to describe lane line geometry and visibility ($\{\alpha_k, \beta_k, \gamma_k\}_{k=1}^{K}$), as well as a line presence probability p_{pr} and a probability distribution $\boldsymbol{p}_{cat}$ for different line categories.

2. Training

In this section, we describe the details regarding the training procedure.

2.1. Initial proposals and Matching

We use several initial line proposals to cover a wide variety of lane geometries. More precisely, the proposals are straight lines with different orientations and different positions in the x-y-plane. After investigations of different set configurations, we found the best set of proposals to be the one with $M = 64$ proposals that is illustrated in Fig. 3.

The matching of ground truth lines to the line proposals is inspired by [12], which choose the unilateral chamfer distance (UCD) as a matching criterion. However, we found that a combination of the unilateral chamfer distance (normalized, thus $UCD \in [0, 1]$) and an orientation cost based on the cosine distance ($CosD \in [0, 1]$) better reflects how well a line proposal $\bar{\boldsymbol{f}}$ resembles a ground truth line described by the set of ground truth points $\mathcal{P}_{GT}$. Thus, the pair-wise matching cost between a proposal with index i (with $i \leq M$) and a ground truth line with index j (with $j \leq M_{GT}$ and M_{GT} the number of ground truth lines) is given as

$$L^{(ij)} = \lambda_{UCD} \cdot UCD(\bar{\boldsymbol{f}}^{(i)}, \mathcal{P}_{GT}^{(j)}) + \tag{1}$$

$$\lambda_{CosD} \cdot CosD(\bar{\boldsymbol{f}}^{(i)}, \mathcal{P}_{GT}^{(j)}), \tag{2}$$

with weights for each cost component λ_{UCD} and λ_{CosD}. Computing the cost between each line proposal and each ground truth line then yields a cost matrix of shape $M \times M_{GT}$. Finally, for each ground truth line we assign the proposals with pair-wise cost under a specified matching threshold $L^{(ij)} < L_{thr}$.

2.2. Losses and ground truth

We provide more details regarding losses and ground truth.

Indicator function for prior regularization. The parallelism loss uses an indicator function $\mathbb{1}_{\boldsymbol{p}}^{(ij)}$ deciding, whether the loss is applied to the point pair consisting of point $\boldsymbol{p}$ on line i and the best matching point in normal direction $\boldsymbol{p}^*$ on line j. The indicator function is defined as

$$\mathbb{1}_{\boldsymbol{p}}^{(ij)} = \begin{cases} 1 & \text{if} \quad OD_{\boldsymbol{p}^*}^{(ij)} < OD_{thr} \text{ and } \sigma^{(ij)} < \sigma_{thr}\,, \\ 0 & \text{else}\,. \end{cases} \tag{3}$$

As Eq. (3) shows, the parallelism criterion holds if two conditions are fulfilled. The first condition $OD_{\boldsymbol{p}^*}^{(ij)} < OD_{thr}$ takes into account the orthogonal distance (OD) of the best matching point $\boldsymbol{p}^*$ on line j to the normal plane spanned by the tangent $\mathbf{T}^{(i)}(t_{\boldsymbol{p}})$ at point $\boldsymbol{p}$ on line i, which is given as

$$OD_{\boldsymbol{p}^*}^{(ij)} = \mathbf{T}^{(i)}(t_{\boldsymbol{p}})^T \cdot \left(\boldsymbol{f}^{(j)}(t_{\boldsymbol{p}^*}) - \boldsymbol{f}^{(i)}(t_{\boldsymbol{p}})\right). \quad (4)$$

Hence, only point pairs are considered for the parallelism loss, which actually lie in opposite normal direction. This is implied by the orthogonal distance having a small enough value, i.e. if the value is lower than a certain threshold OD_{thr}. For instance, if two neighboring lines have different ranges, the non-overlapping range has no neighbor points that have an orthogonal distance smaller than the threshold. Thus, the condition ensures that only point pairs are considered, which are actual neighbors in normal direction.

The second condition $\sigma^{(ij)} < \sigma_{thr}$ guarantees that parallelism is not reinforced for line pairs, which presumably belong to lanes of different orientations, e.g. for merge and split scenarios. The distinction between parallel and non-parallel line pairs can be determined by evaluating the standard deviation $\sigma^{(ij)}$ of the euclidean distances $D_{\boldsymbol{p}}^{(ij)}$ of point pairs of neighboring lines i and j. The standard deviation is defined as

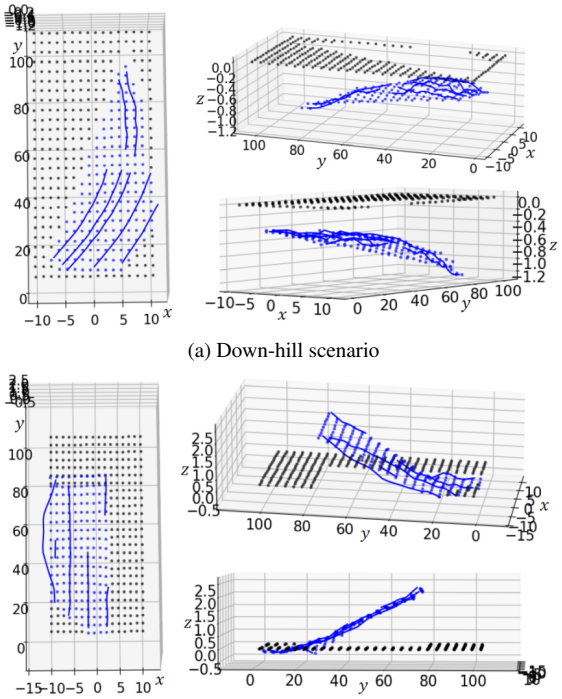
$$\sigma^{(ij)} = \sqrt{\frac{1}{|\mathcal{P}^{(i)}|} \sum_{\boldsymbol{p} \in \mathcal{P}^{(i)}} D_{\boldsymbol{p}}^{(ij)} - \bar{D}^{(ij)}}, \text{ where} \quad (5)$$

$$\bar{D}^{(ij)} = \frac{1}{|\mathcal{P}^{(i)}|} \sum_{\boldsymbol{p} \in \mathcal{P}^{(i)}} D_{\boldsymbol{p}}^{(ij)}, \quad (6)$$

and the euclidean distance for one point pair as $D_{\boldsymbol{p}}^{(ij)} = \left|\left|\boldsymbol{f}^{(i)}(t_{\boldsymbol{p}}) - \boldsymbol{f}^{(j)}(t_{\boldsymbol{p}^*})\right|\right|_2$. For lines of different orientations (as for merging and splitting lines) this standard deviation is rather high and more likely surpasses the threshold σ_{thr} in contrast to lines belonging to the same lane, where $\sigma^{(ij)}$ is rather small.

Ground truth generation for surface loss. For the surface loss computation, height ground truth $\hat{z}_{uv}$ needs to be provided on the $X \times Y$ BEV grid. We approximate this surface ground truth by interpolation of the 3D lane ground truth. For this, we simply compute the convex hull of ground truth lines and interpolate the height value at each cell inside the convex hull. Only cells inside the convex hull are considered for the surface loss, whereas cells outside the convex hull are simply masked out. This is reflected by the indicator function $\mathbb{1}_{uv}$, hence $\mathbb{1}_{uv} = 1$ if cell (u, v) is inside the hull, else $\mathbb{1}_{uv} = 0$. The result of the grid-wise height ground truth generation is visualized in Fig. 4 for an up-hill and a down-hill scenario.

Lane presence and category classification losses. For both classification losses, we apply focal loss [9]. For line presence, which only considers the two classes present and not present, the loss is given as



(a) Down-hill scenario

(b) Up-hill scenario

Figure 4. Examples of the surface ground truth generation. Ground truth lines are visualized as blue lines and height ground truth per cell as blue dots. The black dots correspond to cells outside the convex hull of 3D lines and are not considered for the surface loss.

$$\mathcal{L}_{pr} = -\frac{1}{M} \sum_{i=1}^{M} \left(\hat{p}_{pr}^{(i)} \cdot \left(1 - p_{pr}^{(i)}\right)^{\gamma_f} \cdot \log\left(p_{pr}^{(i)}\right) + \quad (7)$$

$$\left(1 - \hat{p}_{pr}^{(i)}\right) \cdot \left(p_{pr}^{(i)}\right)^{\gamma_f} \cdot \log\left(1 - p_{pr}^{(i)}\right)\Big), \quad (8)$$

with predicted line presence probability $p_{pr}^{(i)}$ for line i and line presence ground truth $\hat{p}_{pr}^{(i)} = \{0, 1\}$. $\gamma_f \geq 0$ denotes the focusing parameter introduced in [9] to handle class imbalance.

The category classification loss is applied for datasets, which provide lane category information in the ground truth. Analogously to Eq. (8), the loss is given as

$$\mathcal{L}_{cat} = -\frac{1}{M} \sum_{i=1}^{M} \frac{1}{C_{cat}} \sum_{c=1}^{C_{cat}} \left(\hat{\boldsymbol{p}}_{cat}^{(i)}[c] \cdot \right. \quad (9)$$

$$\left.\left(1 - \boldsymbol{p}_{cat}^{(i)}[c]\right)^{\gamma_f} \cdot \log\left(\boldsymbol{p}_{cat}^{(i)}[c]\right)\right), \quad (10)$$

with the predicted category probability vector $\boldsymbol{p}_{cat}^{(i)} \in \mathbb{R}^{C_{cat}}$, which represents the categorical distribution for line i, and the ground truth one-hot vector $\hat{\boldsymbol{p}}_{cat}^{(i)} \in \{0, 1\}^{C_{cat}}$. Moreover, $\boldsymbol{p}_{cat}^{(i)}[c]$ denotes the c^{th} entry of the vector $\boldsymbol{p}_{cat}^{(i)}$.

Regression loss. For both, the regression and visibility loss, the curve argument $t_{\boldsymbol{p}}$ has to be determined for a respective point in the ground truth $\boldsymbol{p} \in \mathcal{P}_{GT}$. Since our model learns to predict orthogonal offsets from the assigned line proposal, the points are projected orthogonal onto the line proposal as illustrated in Fig. 5. After having obtained the curve arguments in orthogonal direction, the regression loss for a line proposal i is given as

$$\mathcal{L}_{reg}^{(i)} = \frac{1}{|\mathcal{P}_{GT}^{(i)}|} \sum_{\boldsymbol{p} \in \mathcal{P}_{GT}^{(i)}} \hat{v}_{\boldsymbol{p}}^{(i)} \cdot \left|\left| \boldsymbol{w} \odot \left(\boldsymbol{f}^{(i)}(t_{\boldsymbol{p}}) - \begin{pmatrix} \hat{x}_{\boldsymbol{p}}^{(i)} \\ \hat{y}_{\boldsymbol{p}}^{(i)} \\ \hat{z}_{\boldsymbol{p}}^{(i)} \end{pmatrix} \right) \right|\right|_1 \tag{11}$$

with $\hat{v}_{\boldsymbol{p}}^{(i)}$ the ground truth visibility information and $(\hat{x}_{\boldsymbol{p}}^{(i)}, \hat{y}_{\boldsymbol{p}}^{(i)}, \hat{z}_{\boldsymbol{p}}^{(i)})^T$ the 3D position of a ground truth point $\boldsymbol{p}$ on line i. $\boldsymbol{w} \in \mathbb{R}^3$ is a vector with weighting factors for each 3D component providing for a more balanced regression in each dimension. As shown in Eq. (11) and illustrated in Fig. 5a, only visible points are utilized. The total regression loss for all lines is given as

$$\mathcal{L}_{reg} = \frac{1}{M} \sum_{i=1}^{M} \hat{p}_{pr}^{(i)} \cdot \mathcal{L}_{reg}^{(i)} . \tag{12}$$

For completeness, we also provide the visibility loss for each line as

$$\mathcal{L}_{vis}^{(i)} = -\frac{1}{|\mathcal{P}_{GT}^{(i)}|} \sum_{\boldsymbol{p} \in \mathcal{P}_{GT}^{(i)}} \hat{v}_{\boldsymbol{p}}^{(i)} \cdot \log\left(\sigma\left(v^{(i)}(t_{\boldsymbol{p}})\right)\right) + \tag{13}$$

$$(1 - \hat{v}_{\boldsymbol{p}}^{(i)}) \cdot \log\left(1 - \sigma\left(v^{(i)}(t_{\boldsymbol{p}})\right)\right) . \tag{14}$$

As illustrated in Fig. 5b, all points from the ground truth line are considered. The total visibility loss is then given as

$$\mathcal{L}_{vis} = \frac{1}{M} \sum_{i=1}^{M} \hat{p}_{pr}^{(i)} \cdot \mathcal{L}_{vis}^{(i)} . \tag{15}$$

3. Additional implementation details

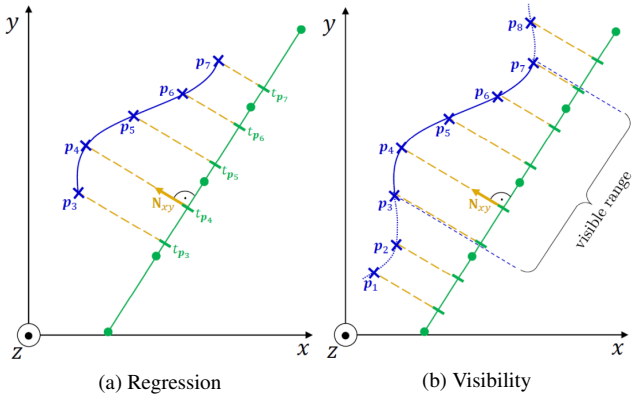
In the following, we provide more implementation details.

3.1. Matching

The weights for the matching cost are $\lambda_{UCD} = 0.5$ and $\lambda_{CosD} = 0.5$, and the distance threshold $L_{thr} = 0.4$.

3.2. Losses

The weights for the different losses are $\lambda_{pr} = 20$, $\lambda_{cat} = 2$, $\lambda_{reg} = 0.5$, $\lambda_{par} = 10$, $\lambda_{sm} = 0.01$, $\lambda_{curv} = 1$, $\lambda_{prior} = 1$, $\lambda_{surf} = 0.1$. The focusing parameter for the classification losses is $\gamma_f = 6.0$ and the vector to weight each dimension for the regression loss is $\boldsymbol{w} = (2, 10, 1)^T$. The thresholds for the indicator function used for the prior losses are $\sigma_{thr} = 2\,\text{m}$ and $OD_{thr} = 1\,\text{m}$ and the thresholds for the maximum curvatures are $\kappa_{xy} = 5$ and $\kappa_z = 0.1$. The set of ground truth points considered for the visibility and regression losses has size $|\mathcal{P}_{GT}| = 20$. For the parallelism and surface smoothness loss we sample $|\mathcal{P}| = 20$ points from the predictions and $|\mathcal{P}| = 100$ points for the curvature loss.



(a) Regression (b) Visibility

Figure 5. Projection of ground truth points $\boldsymbol{p}$ onto line proposal in normal direction to obtain curve arguments $t_{\boldsymbol{p}}$. For regression (a) only visible points are considered (continuous lines), for visibility (b) all points are taken into account, where invisible points are marked with dashed lines.

3.3. Training procedure

In the training, we use Adam optimizer [7], with an initial learning rate of $2 \cdot 10^{-4}$ for OpenLane and 10^{-4} for Apollo. We use a dataset specific scheduler: We train for 30 epochs on OpenLane, where the learning rate is decreased to $5 \cdot 10^{-5}$ after 27 epochs, and for 300 epochs on Apollo, where the learning rate is divided by two every 100 epochs.

3.4. Others

The maximum number of cells used for feature pooling in the detection head is max_cells $= 64$.

4. Additional results

In this section, we provide additional quantitative and qualitative results.

4.1. Ablation studies

Table 2 shows the performance of 3D-SpLineNet [12] on OpenLane300 and the effect of different design adaptations. It is clearly evident that these modifications result in large improvements that were necessary to make the approach applicable to real-world data.

Config	3D-SpLineNet	+BB	+BB+MS	+BB+MS+FP
F1(%)↑	50.9	53.7	58.7	62.9

Table 2. Performance on OpenLane300 of 3D-SpLineNet baseline and architecture adaptations, i.e. larger backbone (BB), multi-scale features (MS) and feature pooling in detection head (FP).

Uniform	**Sampling Rate**	**1**	**3**	**5**	**15**	**27**
	F1-Score(%)↑	15.4	30.9	39.6	48.4	51.1
Surface H.	**Sampling Rate**	**1**	**3**	**5**	**15**	**27**
	F1-Score(%)↑	65.0	65.9	66.6	66.1	66.0

Table 3. Effect of the sampling strategy used in the spatial transformation on OpenLane300. Uniform ray sampling is compared to samples obtained from intersections of rays with surface hypotheses.

In Table 3 we compare two different strategies to draw samples from the camera rays to investigate the effect of using priors in form of surface hypotheses for this component. The samples determine the frustum-like pseudo point cloud in 3D space as described in Sec. 3.3 in the main paper. For the uniform sampling (comparable to [11]), the samples are drawn along the rays with equal step size in the range $[3\,\text{m}, 110\,\text{m}]$ to guarantee that the whole space of interest is covered. We compare this method to our sampling based on prior-incorporated surface hypotheses as proposed and described in the main paper. As shown in Table 3, the performance gaps between the two strategies are significant. This highlights the importance of modeling geometry-aware 3D features by generating samples in the space of interest using knowledge about the surface geometry. The differences in F1-Score for varying sampling rates also imply that a uniform sampling strategy requires high sampling rates to achieve comparable performance. In contrast, using surface hypotheses, lower sampling rates are sufficient which keeps the computational costs lower.

4.2. Quantitative results

In Table 4 we report the detailed evaluation metrics of our best performing LaneCPP model for the different scenarios on OpenLane. We provide geometric errors, as well as F1-Score, precision, recall and categorical accuracy.

Besides, we provide a more detailed evaluation on the Apollo 3D Synthetic dataset on all three test sets as shown in Table 5.

4.3. Qualitative results

We show additional qualitative results on OpenLane in Fig. 6. Considering the top rows, it is clearly evident in all examples that our LaneCPP detects lanes more accurately compared to 3D-SpLineNet, which performs poorly on real-world data. The bottom row shows a direct comparison of LaneCPP and PersFormer. Particularly in curves (Fig. 6a - Fig. 6b) and up- or down-hill scenarios (Fig. 6d - Fig. 6f) our model shows high-quality detections compared to PersFormer. For the intersection scenario (Fig. 6c) with many different line instances, LaneCPP shows overall good results but still leaves room for improvement with respect to geometrical precision. A possible solution to improve the behavior in such cases could be to model lane line relations explicitly to better capture global context as mentioned in our future work section. Moreover, we prove that our model is able to classify line categories accurately as illustrated in the middle row plots.

We further demonstrate the results of our model on Apollo 3D Synthetic illustrated in Fig. 7. As shown, our model achieves accurate detection results in simple scenarios from the Balanced Scenes test set (Fig. 7a - Fig. 7b), in more challenging up- and down-hill scenarios from the Rare Scenes test set (Fig. 7c - Fig. 7d) as well as in case of visual variations (Fig. 7e - Fig. 7f). A very challenging scene is shown in Fig. 7f, where our model manages to capture the overall line structure well but still could be improved slightly with respect to close-range x-errors.

References

[1] Yifeng Bai, Zhirong Chen, Zhangjie Fu, Lang Peng, Pengpeng Liang, and Erkang Cheng. Curveformer: 3d lane detection by curve propagation with curve queries and attention. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2023. 7

[2] Li Chen, Chonghao Sima, Yang Li, Zehan Zheng, Jiajie Xu, Xiangwei Geng, Hongyang Li, Conghui He, Jianping Shi, Yu Qiao, et al. Persformer: 3d lane detection via perspective transformer and the openlane benchmark. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2022. 1, 6, 7, 8, 9, 10

[3] Noa Garnett, Rafi Cohen, Tomer Pe'er, Roee Lahav, and Dan Levi. 3d-lanenet: End-to-end 3d multiple lane detection. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 7

[4] Yuliang Guo, Guang Chen, Peitao Zhao, Weide Zhang, Jinghao Miao, Jingao Wang, and Tae Eun Choe. Gen-lanenet: A generalized and scalable approach for 3d lane detection. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020. 7, 11

[5] Shaofei Huang, Zhenwei Shen, Zehao Huang, Zi han Ding, Jiao Dai, Jizhong Han, Naiyan Wang, and Si Liu. Anchor3dlane: Learning to regress 3d anchors for monocular 3d lane detection. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023. 7

[6] Yujie Jin, Xiangxuan Ren, Fengxiang Chen, and Weidong Zhang. Robust monocular 3d lane detection with dual attention. In *Proc. IEEE International Conf. on Image Processing (ICIP)*, 2021. 7

[7] Diederik P. Kingma and Jimmy Ba. Adam: A method for

Scenario	F1(%)↑	P(%)↑	R(%)↑	Categorical Accuracy(%)↑	X-error (m)↓ near	X-error (m)↓ far	Z-error (m)↓ near	Z-error (m)↓ far
Up & Down	53.6	58.4	49.5	90.0	0.338	0.433	0.122	0.188
Curve	64.4	67.7	61.4	91.1	0.283	0.441	0.075	0.117
Extreme Weather	56.7	63.4	51.2	88.8	0.333	0.253	0.081	0.113
Night	54.9	60.6	50.2	82.9	0.318	0.323	0.104	0.166
Intersection	52.0	56.6	48.1	84.7	0.316	0.343	0.099	0.140
Merge & Split	58.7	63.2	54.8	86.0	0.284	0.330	0.066	0.105
All	60.3	64.7	56.5	87.1	0.264	0.310	0.077	0.117

Table 4. Detailed quantitative evaluation of our LaneCPP for different scenarios on OpenLane [2].

stochastic optimization. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2015. 4

[8] Chenguang Li, Jia Shi, Ya Wang, and Guangliang Cheng. Reconstruct from top view: A 3d lane detection approach based on geometry structure prior. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 7

[9] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017. 3

[10] Ruijin Liu, Dapeng Chen, Tie Liu, Zhiliang Xiong, and Zejian Yuan. Learning to predict 3d lane shape and camera pose from a single image via geometry constraints. In *Proc. of the Conf. on Artificial Intelligence (AAAI)*, 2022. 7

[11] Jonah Philion and Sanja Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020. 5

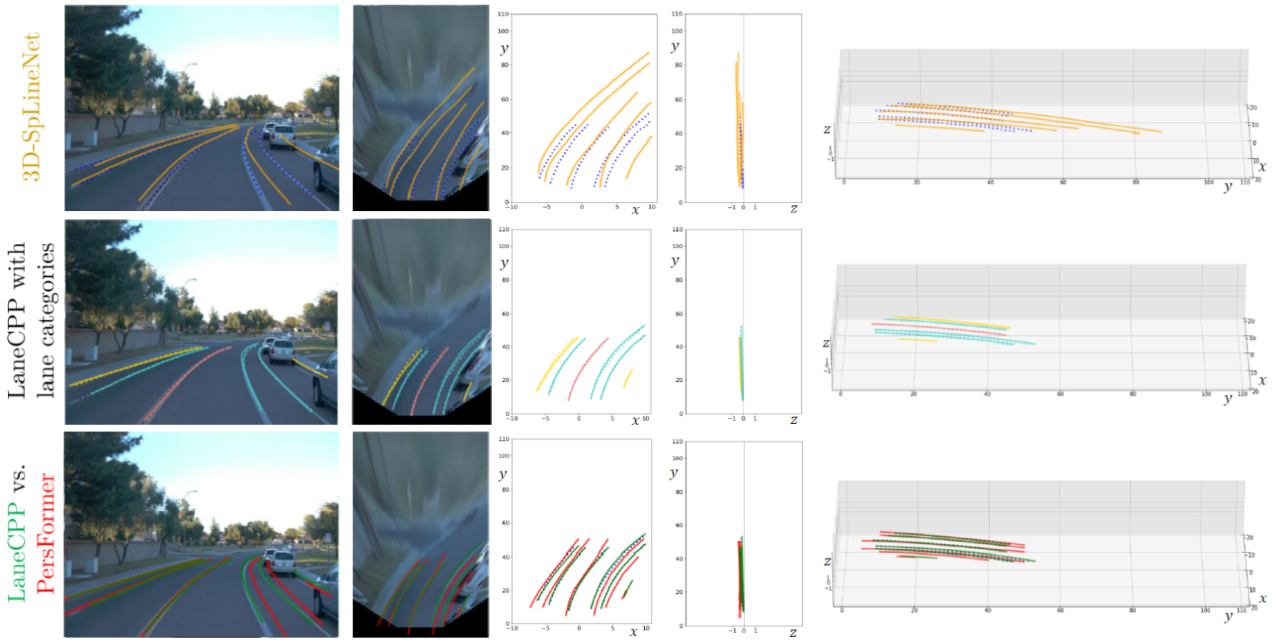
[12] Maximilian Pittner, Alexandru Condurache, and Joel Janai. 3d-splinenet: 3d traffic line detection using parametric spline representations. In *Proc. of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2023. 2, 4, 7

[13] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proc. of the International Conf. on Machine learning (ICML)*, 2019. 1

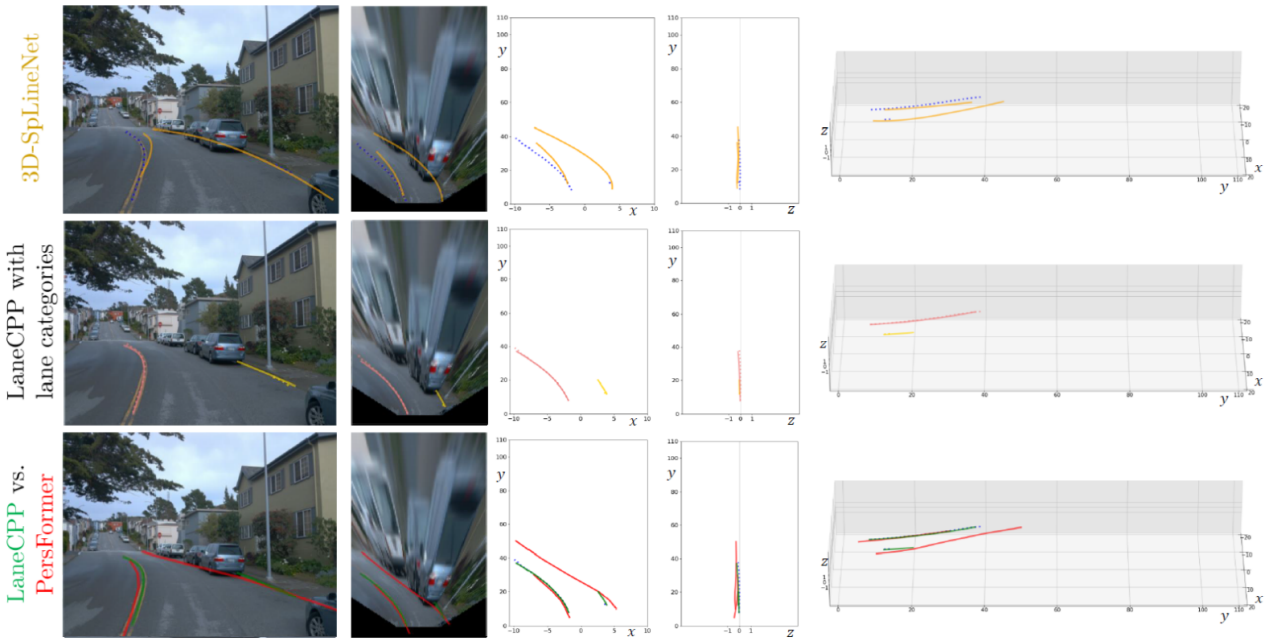
[14] Ruihao Wang, Jian Qin, Kaiying Li, Yaochen Li, Dong Cao, and Jintao Xu. Bev-lanedet: An efficient 3d lane detection based on virtual camera via key-points. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023. 7

Scenario	Method	F1-Score(%)↑	AP(%)↑	X-error (m)↓		Z-error (m)↓	
				near	far	near	far
Balanced Scenes	3D-LaneNet [3]	86.4	89.3	0.068	0.477	0.015	**0.202**
	Gen-LaneNet [4]	88.1	90.1	0.061	0.496	0.012	0.214
	3D-LaneNet (1/att) [6]	91.0	93.2	0.082	0.439	0.011	0.242
	Gen-LaneNet (1/att) [6]	90.3	92.4	0.08	0.473	0.011	0.247
	CLGO [10]	91.9	94.2	0.061	0.361	0.029	0.250
	GP [8]	91.9	93.8	0.049	0.387	**0.008**	0.213
	PersFormer [2]	92.9	–	0.054	0.356	0.010	0.234
	3D-SpLineNet [12]	96.3	98.1	0.037	0.324	0.009	0.213
	CurveFormer [1]	95.8	97.3	0.078	0.326	0.018	0.219
	BEV-LaneDet [14]	96.9	–	**0.016**	**0.242**	0.02	0.216
	Anchor3DLane [5]	95.4	97.1	0.045	0.300	0.016	0.223
	LaneCPP	**97.4**	**99.5**	0.030	0.277	0.011	0.206
Rare Scenes	3D-LaneNet [3]	72.0	74.6	0.166	0.855	0.039	**0.521**
	Gen-LaneNet [4]	78.0	79.0	0.139	0.903	0.030	0.539
	3D-LaneNet (1/att) [6]	84.1	85.8	0.289	0.925	0.025	0.625
	Gen-LaneNet (1/att) [6]	81.7	83.2	0.283	0.915	0.028	0.653
	CLGO [10]	86.1	88.3	0.147	0.735	0.071	0.609
	GP [8]	83.7	85.2	0.126	0.903	0.023	0.625
	PersFormer [2]	87.5	–	0.107	0.782	0.024	0.602
	3D-SpLineNet [12]	92.9	94.8	0.077	0.699	**0.021**	0.562
	CurveFormer [1]	95.6	97.1	0.182	0.737	0.039	0.561
	BEV-LaneDet [14]	**97.6**	–	**0.031**	**0.594**	0.040	0.556
	Anchor3DLane [5]	94.4	95.9	0.082	0.699	0.030	0.580
	LaneCPP	96.2	**98.6**	0.073	0.651	0.023	0.543
Visual Variations	3D-LaneNet [3]	72.5	74.9	0.115	0.601	0.032	0.230
	Gen-LaneNet [4]	85.3	87.2	0.074	0.538	0.015	0.232
	3D-LaneNet (1/att) [6]	85.4	87.4	0.118	0.559	0.018	0.290
	Gen-LaneNet (1/att) [6]	86.8	88.5	0.104	0.544	0.016	0.294
	CLGO [10]	87.3	89.2	0.084	0.464	0.045	0.312
	GP [8]	89.9	92.1	0.060	0.446	**0.011**	0.235
	PersFormer [2]	89.6	–	0.074	0.430	0.015	0.266
	3D-SpLineNet [12]	91.3	93.1	0.069	0.468	0.013	0.248
	CurveFormer [1]	90.8	93.0	0.125	0.410	0.028	0.254
	BEV-LaneDet [14]	**95.0**	–	**0.027**	**0.320**	0.031	0.256
	Anchor3DLane [5]	91.8	92.5	0.047	0.327	0.019	**0.219**
	LaneCPP	90.4	**93.7**	0.054	0.327	0.020	0.222

Table 5. Quantitative evaluation on Apollo 3D Synthetic [4]. **Best performance** and second best are highlighted.

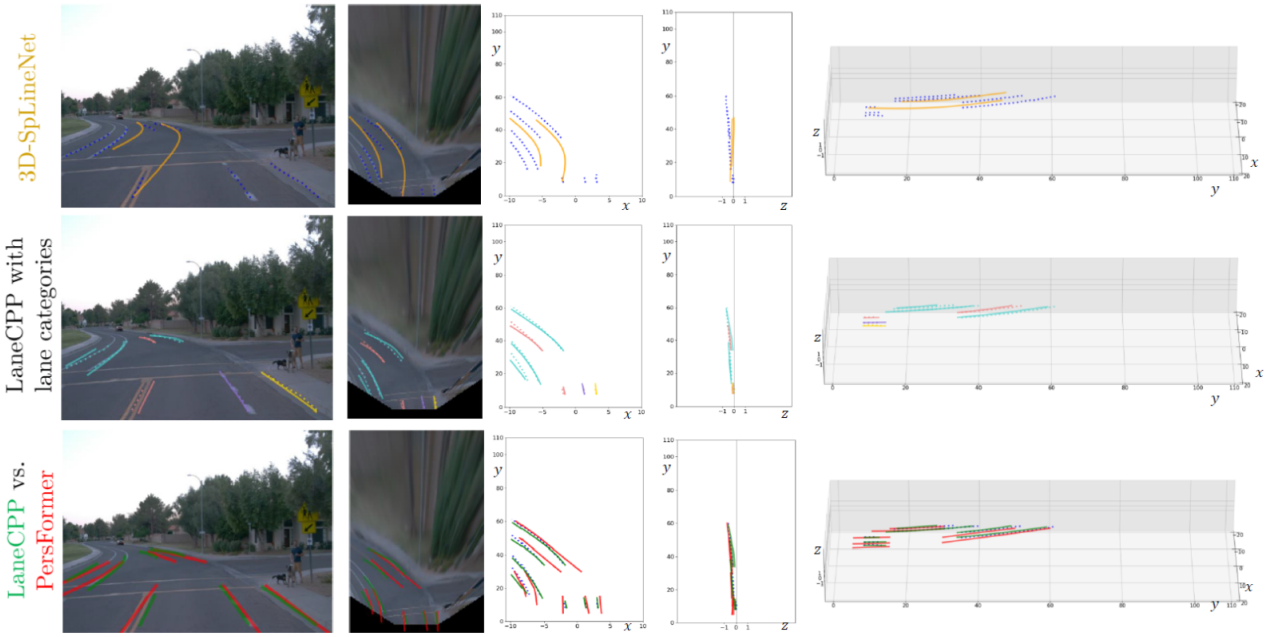


(a)

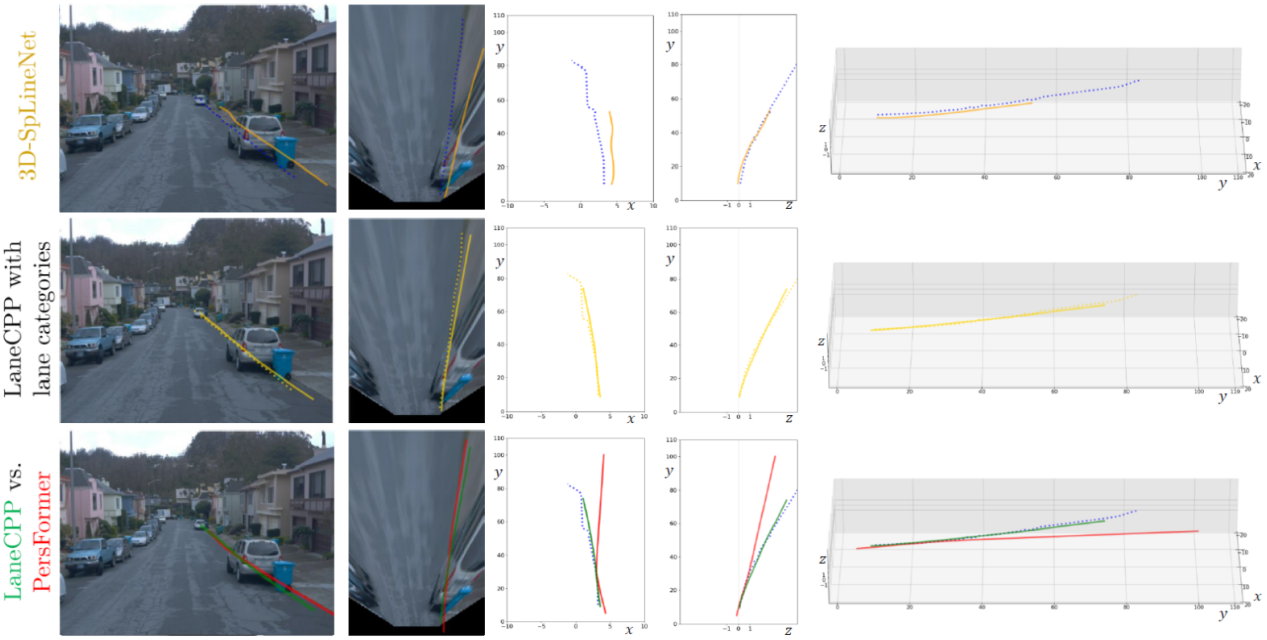


(b)

Figure 6. Additional qualitative evaluation on OpenLane [2] test set (1/3). Top row shows 3D-SpLineNet baseline compared to ground truth. Middle row shows LaneCPP with different lane categories illustrated in different colors and ground truth in dashed lines. Bottom row shows direct comparison of LaneCPP and PersFormer*.

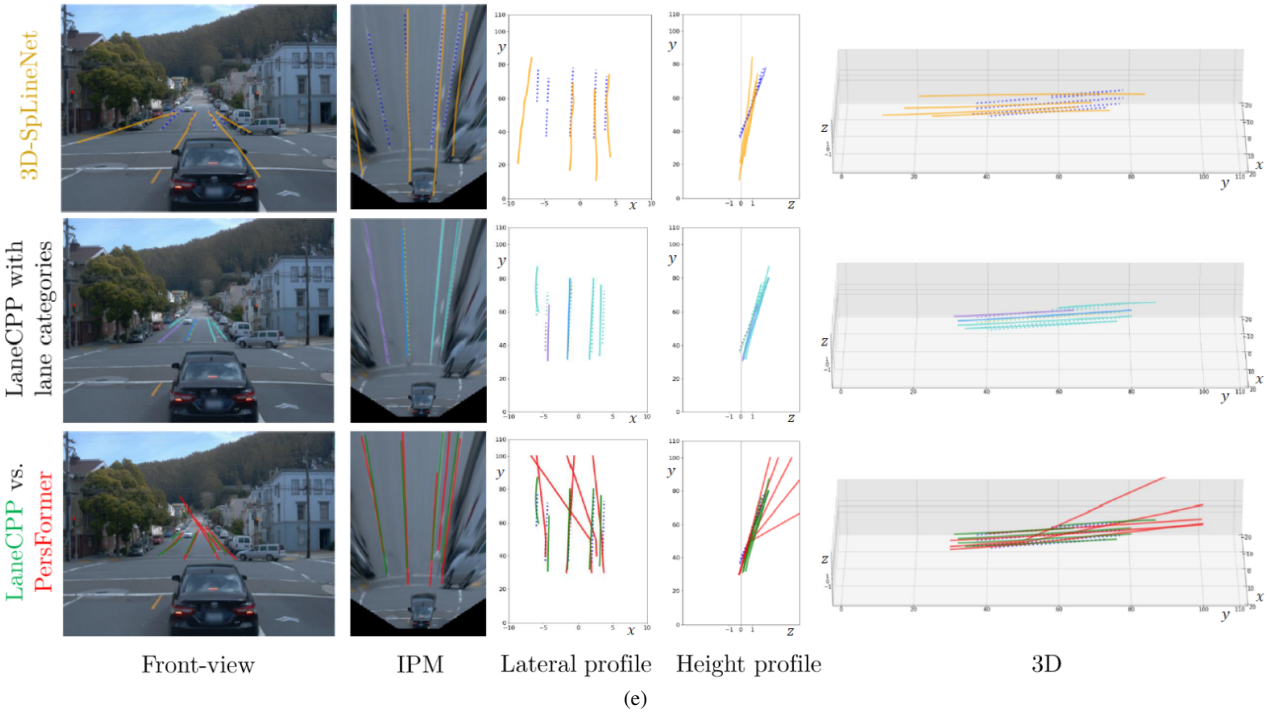


(c)

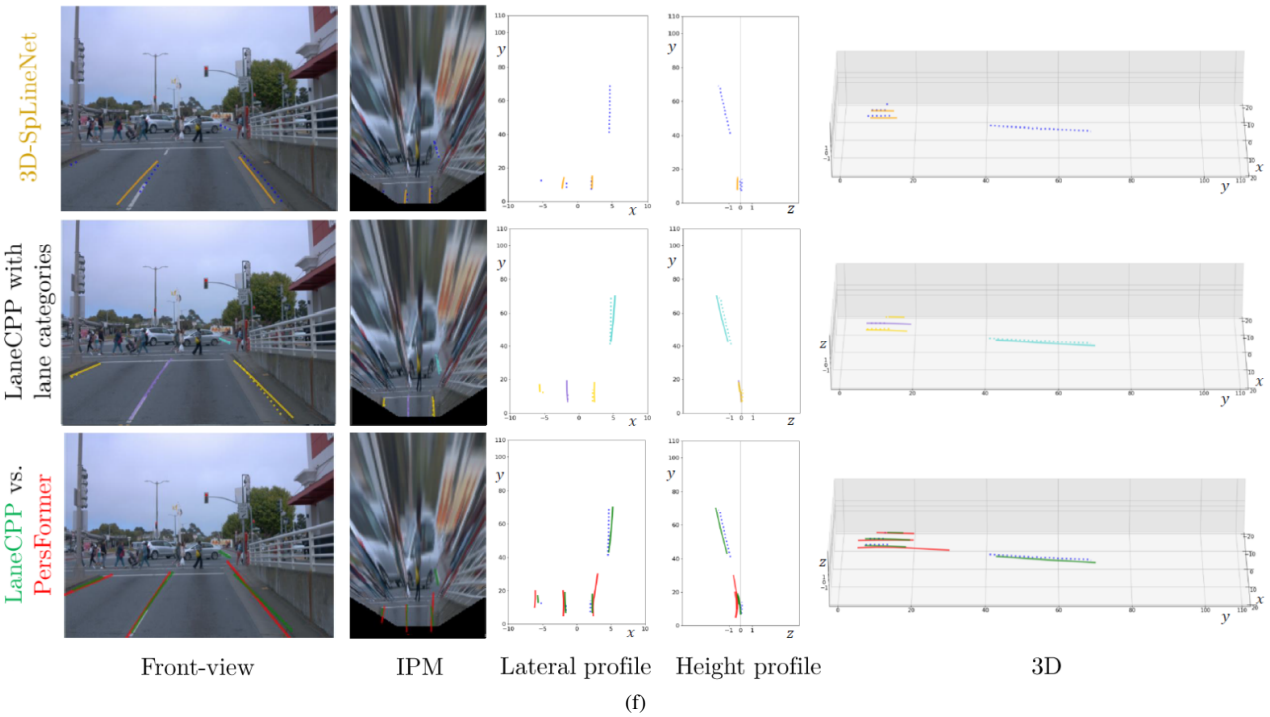


(d)

Figure 6. Additional qualitative evaluation on OpenLane [2] test set (2/3). Top row shows 3D-SpLineNet baseline compared to ground truth. Middle row shows LaneCPP with different lane categories illustrated in different colors and ground truth in dashed lines. Bottom row shows direct comparison of LaneCPP and PersFormer*.



(e)



(f)

Figure 6. Additional qualitative evaluation on OpenLane [2] test set (3/3). Top row shows 3D-SpLineNet baseline compared to ground truth. Middle row shows LaneCPP with different lane categories illustrated in different colors and ground truth in dashed lines. Bottom row shows direct comparison of LaneCPP and PersFormer*.

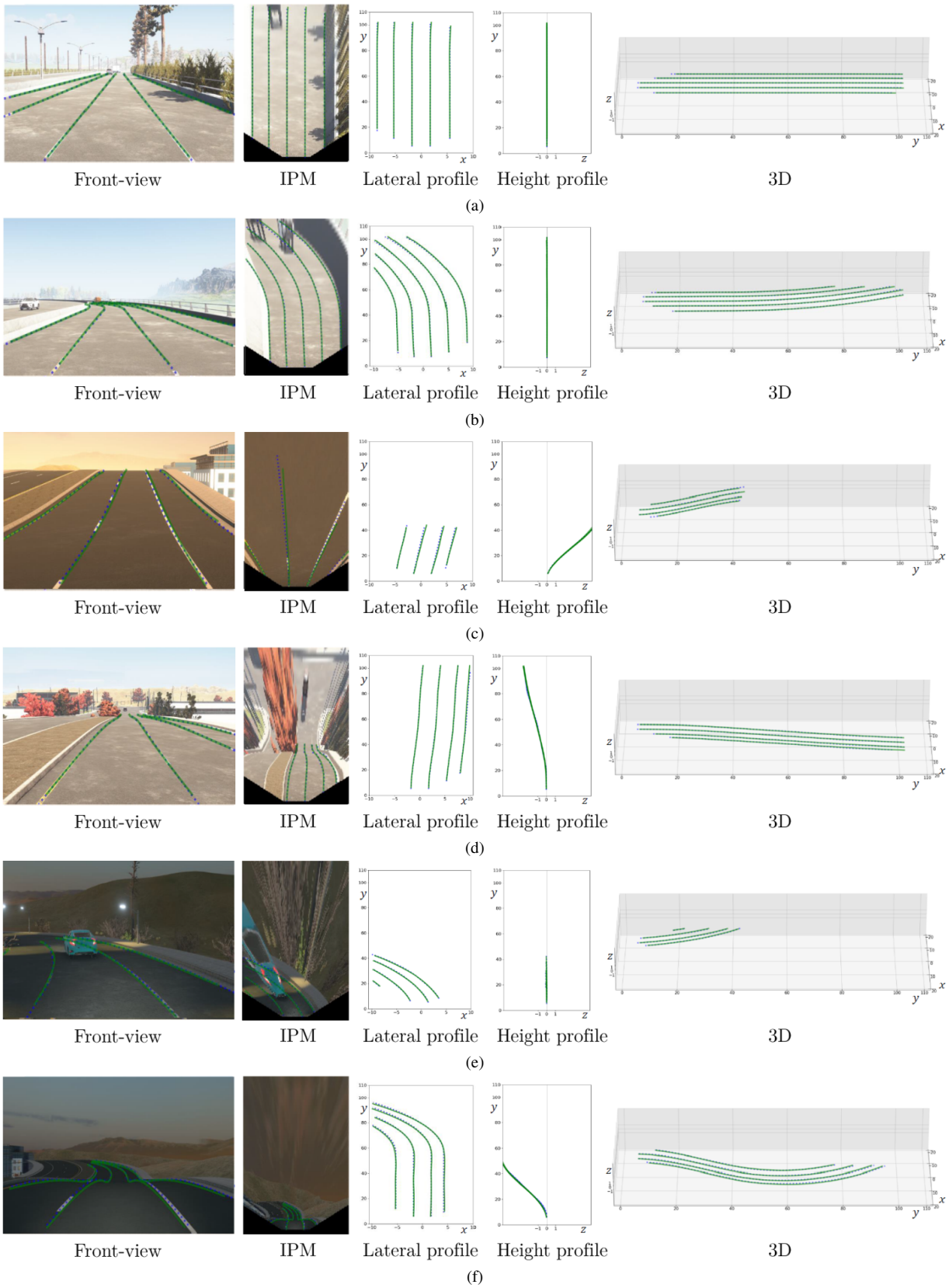


Figure 7. Qualitative evaluation on Apollo 3D Synthetic [4]. Our method is compared to the ground truth visualized dashed.