

# LaneCPP: Continuous 3D Lane Detection using Physical Priors

Maximilian Pittner<sup>1,2</sup>, Joel Janai<sup>1</sup>, Alexandru P. Condurache<sup>1,2</sup>

<sup>1</sup>Bosch Mobility Solutions, Robert Bosch GmbH

<sup>2</sup>Institute of Signal Processing, University of Lübeck

{Maximilian.Pittner, Joel.Janai, AlexandruPaul.Condurache}@de.bosch.com

## Abstract

*Monocular 3D lane detection has become a fundamental problem in the context of autonomous driving, which comprises the tasks of finding the road surface and locating lane markings. One major challenge lies in a flexible but robust line representation capable of modeling complex lane structures, while still avoiding unpredictable behavior. While previous methods rely on fully data-driven approaches, we instead introduce a novel approach LaneCPP that uses a continuous 3D lane detection model leveraging physical prior knowledge about the lane structure and road geometry. While our sophisticated lane model is capable of modeling complex road structures, it also shows robust behavior since physical constraints are incorporated by means of a regularization scheme that can be analytically applied to our parametric representation. Moreover, we incorporate prior knowledge about the road geometry into the 3D feature space by modeling geometry-aware spatial features, guiding the network to learn an internal road surface representation. In our experiments, we show the benefits of our contributions and prove the meaningfulness of using priors to make 3D lane detection more robust. The results show that LaneCPP achieves state-of-the-art performance in terms of F-Score and geometric errors.*

## 1. Introduction

Robust and precise lane detection systems build one of the most essential components in the perception stack of autonomous vehicles. While some approaches utilize LiDAR sensors or multi-sensor setups, the application of monocular cameras has become more popular due to their lower cost and the high-resolution visual representation that provides valuable information to detect lane markings.

In the past, lane detection was mainly treated as a 2D detection task. Deep learning based methods achieved good results by treating the problem as a segmentation task in pixel space [7, 11, 17, 27, 30, 34, 54], used to classify and regress lanes using anchor-based [20, 42] representations,

or as key-points on a grid structure [13, 16, 35, 46]. However, due to the lack of depth information, these 2D representations fail to model lane markings and road geometry in 3D space, which forms an important prerequisite for later functionalities like trajectory planning. Consequently, approaches for monocular 3D lane detection were introduced, which adapted lane representations for the 3D domain by modeling vertical anchors [6, 9] or local segments on a grid [4] in a Birds-Eye-View (BEV) oriented 3D-frame.

A crucial topic for the application of lane detection algorithms in autonomous systems is safety, which requires predictable and robust behavior in any traffic situation. One risk of learning-based methods is the tendency to show unpredictable behavior in cases of rarely observed scenarios. Since obtaining large amounts of data with high-quality annotations is cumbersome and expensive, publicly available 3D datasets are limited in size and accuracy. Hence, they do not reflect the variability of real-world scenarios sufficiently. This makes learning-based models prone to overfitting, and eventually, diminishes predictability.

One common way to deal with such problems is the integration of prior knowledge. Physics provides us a profound understanding of the 3D world, allowing us to make valid assumptions about the lane structure and road surface geometry. Therefore, we introduce physically motivated priors into the lane detection objective to cope with the limited data problem and achieve robust and predictable behavior.

There are certain geometric properties that should generally hold for detected lane lines. For instance, we know that most lines progress parallel to each other, reside on a smooth surface and should not exceed certain thresholds in terms of curvature and slope. However, integrating such assumptions into prevailing discrete representations is not straight forward as strong simplifications are necessary. In contrast, continuous 3D lane representations directly provide parametric curves using polynomials [1, 24] or more sophisticated B-Splines [33]. These allow for analytical computations on the curve function, which enables the integration of such priors into the lane representation. By modeling these priors explicitly instead of learning them from

data, the model can focus its full capacity on learning richer features for the lane detection task.

We can further use physical knowledge about the road geometry to support the model in learning an internal transformation from image features to 3D space. While methods based on Inverse Perspective Mapping (IPM) [4, 6, 9, 18, 24, 33] make false flat-ground assumptions, learning based transformations [1, 2, 47] completely ignore road properties. In contrast, integrating prior knowledge about the road surface allows us to model 3D features geometry-aware and helps the network to focus on the 3D region of interest.

Thus, we propose a novel 3D lane detection approach named LaneCPP that leverages valuable prior knowledge to achieve accurate and robust perception behavior. It introduces a new sophisticated continuous curve representation, which enables us to incorporate physical priors. In addition, we present a spatial transformation component for learning a physically inspired mapping from 2D image to 3D space providing meaningful spatial features.

Our main contributions can be summarized as follows:

- We propose a novel architecture for 3D lane detection from monocular images using a more sophisticated flexible parametric spline-based lane representation.
- We present a way to incorporate priors about lane structure and geometry into our continuous representation.
- We introduce a new way to use prior knowledge about the road surface geometry for learning spatial features.
- We demonstrate the benefits of our contributions in several ablation studies.
- We show state-of-the-art performance of our model.

## 2. Related work

**Different Lane Representations.** An important design choice in deep learning based lane detection is the representation that the network uses to model lane line geometry, which can be categorized as follows: 1) *Pixel-wise* representations, which formulate lane detection as a segmentation problem, were used mainly in 2D methods [7, 11, 17, 27, 30, 34, 52, 54] and were adopted in 3D by SALAD [51] combining line segmentation with depth-prediction. These representations come with high computational load since a large amount of parameters is required. 2) *Grid-based* approaches divide the space into cells and model lanes using local segments [13] or key-points [16, 35, 46]. 3D-LaneNet+ [4] suggests to use local line-segments and BEV-LaneDet [47] defines key-points on a BEV grid representation. Both depend on the grid resolution and require costly post-processing to obtain lines. 3) *Anchor-based* representations [20, 41, 42, 53] model lines as straight anchors with positional offsets at predefined locations. They are widely used in 3D detection approaches including 3D-LaneNet [6] and Gen-LaneNet [9], which use vertical anchors in the top-view, and An-

chor3DLane [12], introducing anchor projection with iterative regression. Similar to grid-based representations, it requires subsequent curve-fitting to obtain smooth lines.

4) *Continuous curve* representations [5, 23, 25, 44, 45] instead directly model smooth curves without requiring costly post-processing. While CLGO [24] and CurveFormer [1] use simple polynomials, 3D-SpLineNet [33] proposes B-Splines [3]. Since B-Splines offer local control over curve segments, they are compatible to model complex shapes with low-degree basis functions, while polynomials and Bézier curves show global dependence and thus require higher degrees causing expensive computation. Although 3D-SpLineNet achieves superior detection performance on synthetic data, it unfortunately lacks flexibility as the curve formulation is limited to monotonically progressing lanes, making it hardly applicable to real-world data. To resolve this issue, we propose a more flexible representation based on actual 3D B-Splines. In contrast to discrete grids and anchors, continuous representation even allow us to integrate prior knowledge in an analytical manner.

**Geometry Priors.** Several approaches suggest to incorporate prior knowledge into learning-based methods, e.g. by integrating invariance into the model architecture [36, 37] or task-specific transformations as for trajectory planning [10, 48, 50]. In the field of lane detection, line parallelism has been formulated as a hard constraint to resolve depth ambiguity and determine camera parameters [28, 49]. Deep declarative networks [8] offer a general framework to incorporate arbitrary properties as constraints, by solving a constrained optimization problem in the forward pass. While such methods are appropriate when hard constraints must be enforced, our goal is rather to guide the network in learning typical geometric lane properties by formulating soft constraints in a regularization objective. Such a regularization only affects training and does not require resolving an optimization problem in the forward pass, and thus, comes without additional computational cost during inference. Following this paradigm, SGNet [25, 41] proposes to penalize the deviation of lateral distance from a constant lane width in the IPM warped top-view, but ignores that the property does not hold for lines deviating from the ground plane. GP [18] presents a parallelism loss that enforces constant distance between nearest neighbors locally, which depends on the number of anchor points. In contrast, our method presents a way to learn parallelism globally and independent of resolutions of discrete lane representations. We propose an elegant way to learn parallelism as well as other geometry priors using analytical formulations of tangents and normals, which are well-defined on our continuous spline representation.

**Leveraging 3D Features.** An important model component consists in the extraction of 3D features, encoding valuable information to detect lanes along the road surface.

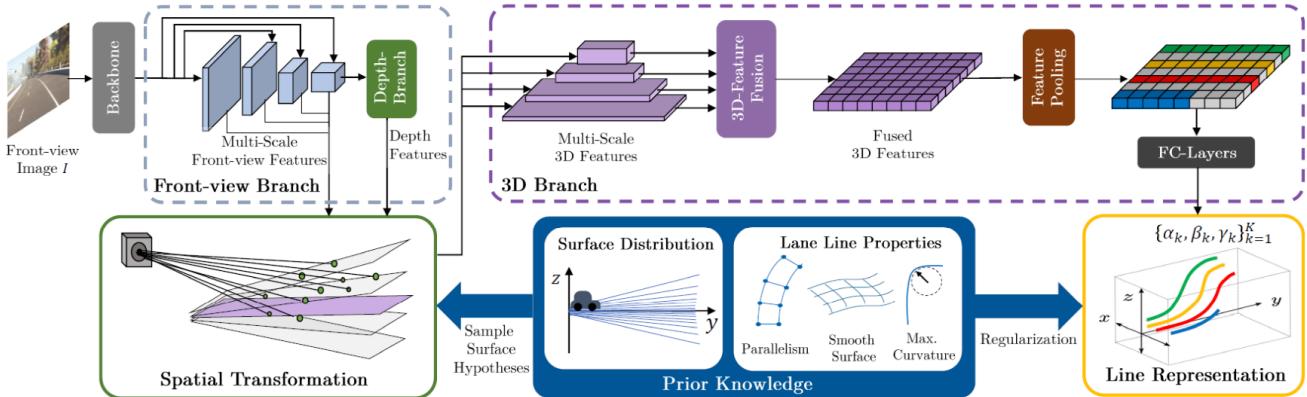


Figure 1. Our approach: First, front-view image  $I$  is propagated through the backbone extracting multi-scale feature maps. These are transformed to 3D using our spatial transformation and then fused to obtain a single 3D feature map. Feature pooling is applied to obtain features for each line proposal that are propagated through fully connected layers to obtain the parameters for our line representation. Finally, prior knowledge is exploited to regularize the lane representation and to produce surface hypotheses for the spatial transformation.

While some works predict 3D lanes directly from the front-view, e.g. by utilizing pixel-wise depth estimation [51] or 3D anchor-projection mechanisms [12], prevalent methods employ an intermediate 3D or BEV feature representation with an internal transformation from the front-view to the 3D space. 3D-LaneNet [6] proposes to utilize IPM [26] to project front-view features to a flat road plane due to the spatial correlation between the warped top-view image and 3D lane geometry and was adopted in several other works [4, 9, 18, 24, 33]. However, IPM causes visual distortions in the top-view representation when the flat road assumption is violated. In related fields like BEV semantic segmentation, BEV transformations are learned via Multi-Layer-Perceptrons (MLPs) [19, 29], depth prediction [32, 38, 39] or transformer-based attention mechanisms [21, 31, 40]. In 3D lane detection, PersFormer [2] utilizes attention between front- and top-view, CurveFormer [1] introduces dynamic 3D anchors that model queries as parametric curves and BEV-LaneDet [47] uses MLPs for the spatial transformations. However, these learned transformations do not necessarily provide a 3D feature representation since they are not guided by valuable priors about the road surface geometry, which potentially results in unforeseen behavior for out-of-distribution data. Our approach instead aims for carefully modeling a geometry-aware feature space using a depth classification method inspired by [32] that exploits knowledge about the distribution of the road surface.

### 3. Methodology

The following section describes our 3D lane detection approach. An overview of the overall architecture is described and illustrated in Fig. 1. The main focus lies on our continuous 3D lane line representation, our regularization mechanism using physical priors and our prior-based spatial trans-

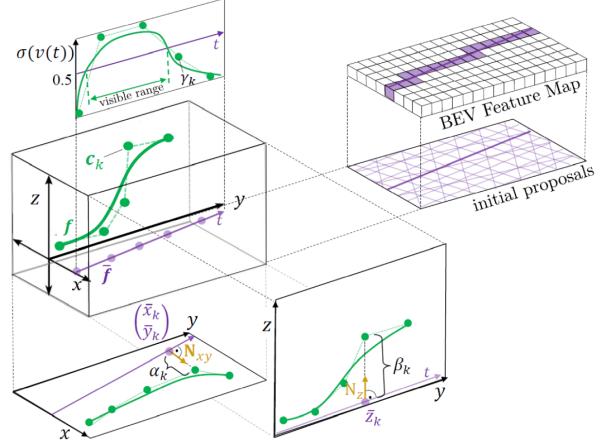


Figure 2. Our 3D lane line representation: For each proposal  $\bar{f}$  (purple lines), line geometry is described by 3D B-Splines with control points  $c_k$  (green dots). Each control point is determined by the offsets  $\alpha_k$ ,  $\beta_k$  from the control points of the initial proposal in normal direction (orange vectors). Additionally, visibility  $v(t)$  is modeled by splines with 1D control points  $\gamma_k$ .

formation module, which we explain in the following.

#### 3.1. Lane line representation

Inspired by prior work in 3D lane detection [33], we leverage the benefits of continuous representations and employ a parametric model based on B-Splines. However, modeling only lateral ( $x$ -) and vertical ( $z$ -) components with spline-based functions (as done in previous approaches) is limited to lanes that merely progress along the longitudinal ( $y$ -) direction. Instead, we propose the first full 3D lane line representation modeling each component ( $x$ ,  $y$ ,  $z$ ) such that we

obtain

$$\mathbf{f}(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix} = \sum_{k=1}^K \mathbf{c}_k \cdot B_{k,d}(t) \quad (1)$$

with curve argument  $t \in [0, 1]$  and  $K$  control points  $\mathbf{c}_k = (x_k, y_k, z_k)^T$ . Each control point  $\mathbf{c}_k$  weights the respective basis function  $B_{k,d}(t)$  (recursive polynomials of degree  $d$ ) controlling the curve shape.

Due to the ambiguity of curves using 3D B-Splines (the same spline curve can be described by different configurations of its control points), regressing all three dimensions per control point results in strong overfitting during training. We resolve this issue by limiting the degrees of freedom per control point to two and constraining the control points deflection to one direction in the  $x$ - $y$ -plane and one direction in the  $y$ - $z$ -plane as illustrated in Fig. 2. More precisely, the degrees of freedom per control point are specified by the directions of the normals  $\mathbf{N}_{xy}$  and  $\mathbf{N}_z$  of an initial curve proposal  $\bar{\mathbf{f}}$  with control points  $\bar{\mathbf{c}}_k = (\bar{x}_k, \bar{y}_k, \bar{z}_k)^T$ . The control points are then defined as

$$\mathbf{c}_k = \begin{pmatrix} x_k \\ y_k \\ z_k \end{pmatrix} = \begin{pmatrix} \bar{x}_k + \mathbf{N}_x \cdot \alpha_k \\ \bar{y}_k + \mathbf{N}_y \cdot \alpha_k \\ \bar{z}_k + \mathbf{N}_z \cdot \beta_k \end{pmatrix}, \quad (2)$$

where  $\mathbf{N}_x, \mathbf{N}_y$  describe the  $x$ - and  $y$ -component of the normal vector  $\mathbf{N}_{xy}$  in the  $x$ - $y$ -plane. As shown in Eq. (2) and illustrated in Fig. 2, modeling splines as deflections in normal direction of its underlying initial line proposal only requires two parameters  $\alpha_k, \beta_k$  per control point to describe the 3D shape. We use a wide variety of orientations for the initial proposals  $\bar{\mathbf{f}}$  (see Fig. 2), which allows us to detect any kind of lines with this formulation. More details about the initial proposals are provided in the supplementary.

While [33] models the curve range using start- and endpoints that are learned by means of regression, we instead propose to model visibility<sup>1</sup> using a continuous representation  $v(t)$  and treat the visibility estimation as a classification problem. We obtain probability values applying sigmoid activation and consider  $\sigma(v(t)) > 0.5$  the visible range. While in theory any kind of function can be utilized, we found that B-Splines with the same configuration as  $\mathbf{f}(t)$  are well-suited and introduce spline control points  $\gamma_k$  defining the shape of  $v(t)$ .

Eventually, binary cross-entropy is used as a classification loss to learn visibility

$$\mathcal{L}_{vis} = -\frac{1}{|\mathcal{P}_{GT}|} \sum_{\mathbf{p} \in \mathcal{P}_{GT}} \hat{v}_{\mathbf{p}} \cdot \log(\sigma(v(t_{\mathbf{p}}))) + \quad (3)$$

$$(1 - \hat{v}_{\mathbf{p}}) \cdot \log(1 - \sigma(v(t_{\mathbf{p}}))), \quad (4)$$

<sup>1</sup>For the concept of visibility, we follow the prevailing definition from the literature [2, 9].

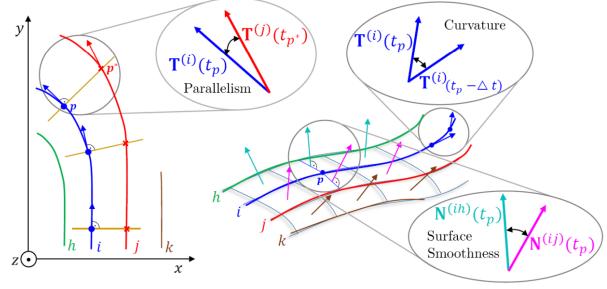


Figure 3. Illustration of different priors expressed by line tangents and surface normals.

where  $\mathcal{P}_{GT}$  denotes the ground truth set of points,  $\hat{v}_{\mathbf{p}} \in \{0, 1\}$  the ground truth visibility for point  $\mathbf{p}$ .  $t_{\mathbf{p}}$  represents the respective curve argument obtained by orthogonal projection of  $\mathbf{p}$  onto the underlying line proposal.

### 3.2. Regularization using physical priors

In this section, we describe our regularization method to integrate prior knowledge about lane structure and surface geometry into our parametric line representation (see Fig. 3).

**Line parallelism.** In order to reinforce parallel lines, the tangents at point pairs located in opposite normal direction on neighboring lines must be similar (see Fig. 3 left). We realize this by penalizing the cosine distance of the unit tangents  $\mathbf{T}(t)$  on neighboring lines  $i$  and  $j$  for normal point pairs. More precisely, for each point  $\mathbf{p} \in \mathcal{P}^{(i)}$  on line  $i$  we select the normal pair point  $\mathbf{p}^*$  on neighbor line  $j$  that minimizes the distance to the normal plane, which is defined by the plane equation  $\mathbf{T}^{(i)}(t)^T \cdot ((x, y, z)^T - \mathbf{f}^{(i)}(t)) = 0$ . In Fig. 3 the normal planes are visualized in a 2D top-view as lines (orange) for simplicity. Hence the respective curve argument  $t_{\mathbf{p}^*}$  for point  $\mathbf{p}^*$  on line  $j$  is given as

$$t_{\mathbf{p}^*} = \operatorname{argmin}_{\mathbf{p}' \in \mathcal{P}^{(j)}} \mathbf{T}^{(i)}(t_{\mathbf{p}})^T \cdot (\mathbf{f}^{(j)}(t_{\mathbf{p}'}) - \mathbf{f}^{(i)}(t_{\mathbf{p}})), \quad (5)$$

where  $\mathcal{P}^{(j)}$  denotes the points on line  $j$ . While in theory Eq. (5) can be solved analytically, the simpler way is to sample the set of points  $\mathcal{P}^{(j)}$  instead. (Note that our continuous representation allows us to choose high sampling rates without losing precision as no interpolation is required.)

With the normal point pairs, we define the parallelism loss for a neighbor line pair based on the cosine distance of their tangents as

$$\mathcal{L}_{par}^{(ij)} = \frac{\mathbb{1}_{\mathbf{p}}^{(ij)}}{|\mathcal{P}^{(i)}|} \cdot \sum_{\mathbf{p} \in \mathcal{P}^{(i)}} 1 - (\mathbf{T}^{(i)}(t_{\mathbf{p}}))^T \cdot \mathbf{T}^{(j)}(t_{\mathbf{p}^*}). \quad (6)$$

Since the criterion of line parallelism should not hold for all normal point pairs of neighboring lines (e.g. merging or splitting lines),  $\mathbb{1}_{\mathbf{p}}^{(ij)} \in \{0, 1\}$  represents the indicator

function determining whether the parallelism loss is applied to the point pair. More precisely, the function ensures that only the overlapping range of neighboring lines is taken into account. Furthermore, it determines whether the line pair should be considered as a parallel pair based on the standard deviation of euclidean distances between normal point pairs, i.e. high deviations indicate that the line pair might belong to a merge or split structure. In our experiments, we achieve state-of-the-art performance on test sets containing merges and splits, proving that our model is also capable of learning non-parallel line pairs using this indicator function.

**Surface smoothness.** Since the lines reside on a smooth road, the surface normals of neighboring lanes should be similar. Analogously to  $\mathcal{L}_{par}$ , we express this with the cosine distance between surface normals  $\mathbf{N}^{(ih)}$  and  $\mathbf{N}^{(ij)}$  as

$$\mathcal{L}_{sm}^{(i)} = \frac{\mathbb{1}_p^{(hij)}}{|\mathcal{P}^{(i)}|} \cdot \sum_{p \in \mathcal{P}^{(i)}} 1 - (\mathbf{N}^{(ih)}(t_p))^T \cdot \mathbf{N}^{(ij)}(t_p), \quad (7)$$

with indicator function  $\mathbb{1}_p^{(hij)}$ . The surface normal between line  $i$  and left neighbor line  $h$  at point  $p$  can be expressed as the cross product of the tangent on line  $i$  and the normalized connection vector between lines  $i$  and  $h$ , hence  $\mathbf{N}^{(ih)}(t_p) = \mathbf{T}^{(i)}(t_p) \times \frac{\mathbf{f}^{(h)}(t_{p^*}) - \mathbf{f}^{(i)}(t_p)}{||\mathbf{f}^{(h)}(t_{p^*}) - \mathbf{f}^{(i)}(t_p)||}$ . For the normal between line  $i$  and right neighbor  $j$  the sign is flipped to obtain upwards pointing normal vectors.

**Curvature.** We determine lane curvature by computing the second order derivatives as the difference of tangents at consecutive points divided by their euclidean distance as  $\mathbf{T}'(t_p) = \frac{\mathbf{T}(t_p) - \mathbf{T}(t_p - \Delta t)}{||\mathbf{f}(t_p) - \mathbf{f}(t_p - \Delta t)||}$ . The maximum curvature in  $x$ - $y$ -plane (inverse curve radius) and in  $z$  (rate of slope change) have very different value ranges and are therefore restricted by different limits. Hence, we define the two thresholds  $\kappa_{xy}$  and  $\kappa_z$  and formulate the curvature loss on line  $i$  as

$$\mathcal{L}_{curv}^{(i)} = \frac{1}{|\mathcal{P}^{(i)}|} \cdot \sum_{p \in \mathcal{P}^{(i)}} \max(\mathbf{T}'_{xy}^{(i)}(t_p), \kappa_{xy}) \quad (8)$$

$$+ \max(\mathbf{T}'_z^{(i)}(t_p), \kappa_z). \quad (9)$$

Finally, the prior regularization loss is given as

$$\mathcal{L}_{prior} = \sum_{i=1}^M \lambda_{sm} \mathcal{L}_{sm}^{(i)} + \lambda_{curv} \mathcal{L}_{curv}^{(i)} + \sum_{j=1}^N \lambda_{par} \mathcal{L}_{par}^{(ij)}, \quad (10)$$

with individual weights  $\lambda_{par}, \lambda_{sm}, \lambda_{curv}$ . Note that all these properties are expressible by means of tangents and normals, which can be computed analytically on our parametric representation in continuous space. Consequently, minimization of the herein introduced prior losses does not depend on numerical approximations as is the case for anchor-, grid- or key-point representations.

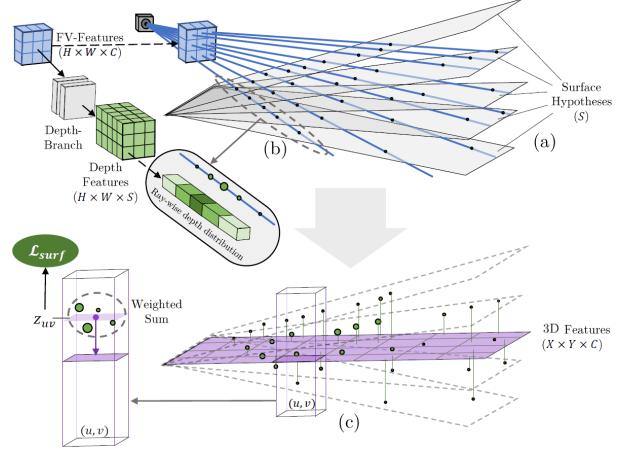


Figure 4. Our proposed spatial transformation module. First, several road surface hypotheses are defined (a) to which front-view features are lifted (b) and weighted according to the predicted depth distribution. Afterwards, point features are aggregated in a weighted manner to obtain the 3D feature map (c).

### 3.3. Spatial transformation

In this section, we describe our spatial transformation (shown in Fig. 4) that is leveraging valuable physical knowledge about surface geometry. We know that the road surface typically shows small deviations from the ground level ( $z = 0$ ) in the near-range and stronger deviations in the far-range. Based on this knowledge, we sample ground surface hypotheses that reflect the distribution of the road surface height profile (Fig. 4a). While in theory different types of surface functions could be utilized as hypotheses, we decide to merely rely on planes, since this facilitates the computation of ray intersections described in the following step.

Next, the multi-scale front-view feature maps extracted by the backbone are lifted to 3D space (Fig. 4b). Our approach is inspired by [32], where front-view features are spreading along rays throughout the space of the road surface. These rays intersect with the surface hypotheses at different depths spanning a frustum-like point cloud in 3D space, where each point is affiliated with a  $C$ -dimensional feature vector and additionally attached with its height value  $z$ , hence, each point in the cloud has dimension  $(C + 1)$ . The front-view feature map is propagated through a depth branch with a channel-wise softmax applied to obtain a categorical distribution for each ray, resulting in a tensor of size  $H \times W \times S$ , where  $H, W$  denote height and width and channel size  $S$  the number of surface hypotheses.

In order to aggregate the information in 3D space, a BEV grid of size  $X \times Y$  is defined. Features from points mapping to the same grid cell are weighted by the categorical depth distribution for the respective ray and accumulated in terms of a weighted sum (Fig. 4c). Since the  $z$ -component

Priors	F1(%)↑	X-near(m)↓	X-far(m)↓	Z-near(m)↓	Z-far(m)↓
None	65.0	0.316	0.384	0.106	0.153
Par.	66.2	0.291	0.373	0.103	0.150
Surf.	65.8	0.320	0.356	0.103	0.144
Curv.	66.7	0.322	0.366	0.105	0.146
Comb.	66.7	0.301	0.359	0.103	0.144

Table 1. Effect of different prior losses on OpenLane300.

# Surface Hypotheses	1	3	5	15	27
F1-Score(%)↑	65.0	65.9	<b>66.6</b>	66.1	66.0

Table 2. Effect of the surface hypotheses on OpenLane300.

Lane Rep.	Prior Reg.	Spatial T.	F1(%)↑	Gain(%)
✓			62.9	(baseline)
✓			65.0	+2.1
✓	✓		66.7	+3.8
✓		✓	66.6	+3.7
✓	✓	✓	<b>66.9</b>	+4.0

Table 3. Performance gain for different contributions on OpenLane300 using our novel **Lane Representation**, **Prior Regularization** and **Spatial Transformation** instead of IPM.

of the points is also combined by a weighted sum, the value  $z_{uv}$  can be interpreted as the height value of the surface for grid cell  $(u, v)$ . We guide the model in learning the real surface and prevent it from learning an arbitrary mapping by introducing a simple grid-based regression loss as

$$\mathcal{L}_{surf} = \frac{1}{X \cdot Y} \sum_{(u,v) \in X \times Y} \mathbb{1}_{uv} \cdot \|z_{uv} - \hat{z}_{uv}\|_1, \quad (11)$$

with  $\mathbb{1}_{uv}$  indicating whether surface ground truth  $\hat{z}_{uv}$  is available for cell  $(u, v)$ . The height ground truth is obtained by interpolation of the 3D lane annotations at cell locations.

### 3.4. Loss functions

The overall loss used during training is given as the weighted sum of loss components

$$\mathcal{L} = \lambda_{pr} \mathcal{L}_{pr} + \lambda_{cat} \mathcal{L}_{cat} + \lambda_{reg} \mathcal{L}_{reg} + \quad (12)$$

$$\lambda_{vis} \mathcal{L}_{vis} + \lambda_{prior} \mathcal{L}_{prior} + \lambda_{surf} \mathcal{L}_{surf}. \quad (13)$$

We use focal loss [22] for lane presence  $\mathcal{L}_{pr}$  and category classification  $\mathcal{L}_{cat}$ . For the regression loss  $\mathcal{L}_{reg}$ , we adapt the formulation of [33] to three instead of two dimensions. More details are provided in the supplementary.

## 4. Experiments

We first describe our experimental setup and then analyze our approach on two 3D lane datasets.

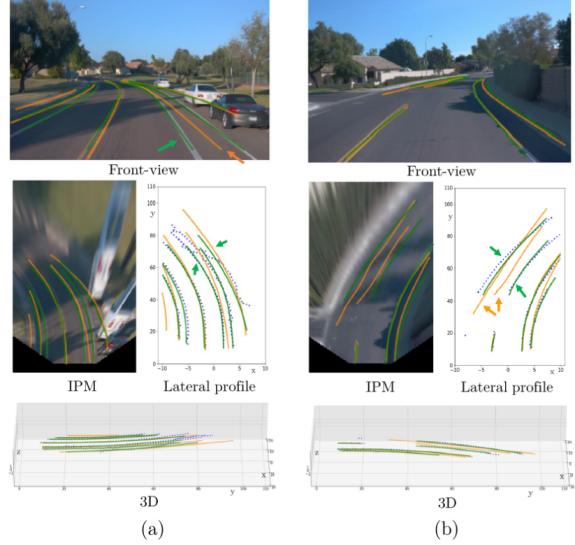


Figure 5. Qualitative comparison of our model trained with prior regularization to the same model without regularization both trained on OpenLane300 with main differences highlighted by arrows. As a reference ground truth lines are visualized dashed.

### 4.1. Experimental setup

We evaluate our method on two different datasets: OpenLane and Apollo 3D Synthetic - both containing 3D lane ground truth as well as camera parameters per frame.

**OpenLane** [2] is a real-world dataset containing 150,000 images in the training and 40,000 in the test set from 1000 different sequences. In order to evaluate different driving scenarios the test set is divided into different situations, namely *Up & Down*, *Curve*, *Extreme Weather*, *Night*, *Intersection* and *Merge & Split*. For ablation studies we use the smaller version OpenLane300 including 300 sequences.

**Apollo 3D Synthetic** [9] is a small synthetic dataset, consisting of only 10,500 examples from rather simple scenarios of highway, urban and rural environments. The data is split into three subsets, (1) *Standard* (simple) scenarios, (2) *Rare Scenes* and (3) *Visual Variations*.

**Evaluation metrics.** For the quantitative evaluation both datasets utilize the evaluation scheme proposed in [9].

It evaluates the euclidean distance at uniformly distributed points in the range of 0-100 m along the  $y$ -direction. Based on the mean distance and range, *F1-Score* is computed, as well as the mean  $x$ - and  $z$ -errors in *near*- (0-40 m) and *far-range* (40-100 m) to evaluate geometric accuracy.

**Baseline.** Our approach builds up on 3D-SpLineNet. Since it was applied on synthetic data only, it showed poor performance on real data. We applied some straightforward design adaptations - e.g. larger backbone, multi-scale features (see supplementary) - and use this modified 3D-SpLineNet as our baseline (first row Table 3).

Method	F1-Score(%)↑	X-error near(m)↓	X-error far(m)↓	Z-error near(m)↓	Z-error far(m)↓	F1-Score(%) per Scenario ↑					
	U&D	C	EW	N	I	M&S					
3D-LaneNet [6]	44.1	0.479	0.572	0.367	0.443	40.8	46.5	47.5	41.5	32.1	41.7
Gen-LaneNet [9]	32.3	0.591	0.684	0.411	0.521	25.4	33.5	28.1	18.7	21.4	31.0
PersFormer [2]	50.5	0.485	0.553	0.364	0.431	42.4	55.6	48.6	46.6	40.0	50.7
PersFormer* [2]	53.1	0.361	0.328	0.124	0.129	46.8	58.7	54.0	48.4	41.4	52.5
CurveFormer [1]	50.5	0.340	0.772	0.207	0.651	45.2	56.6	49.7	49.1	42.9	45.4
BEV-LaneDet [47]	58.4	0.309	0.659	0.244	0.631	48.7	63.1	53.4	53.4	50.3	53.7
Anchor3DLane [12]	53.7	0.276	0.311	0.107	0.138	46.7	57.2	52.5	47.8	45.4	51.2
Anchor3DLane-T [12]	54.3	0.275	0.310	0.105	0.135	47.2	58.0	52.7	48.7	45.8	51.7
LaneCPP (Ours)	<b>60.3</b>	<b>0.264</b>	<b>0.310</b>	<b>0.077</b>	<b>0.117</b>	<b>53.6</b>	<b>64.4</b>	<b>56.7</b>	<b>54.9</b>	<b>52.0</b>	<b>58.7</b>

Table 4. Quantitative comparison on OpenLane [2]. **Best performance** and **second best** are highlighted. The scenario categories are Up and Down (U&D), Curve (C), Extreme Weather (EW), Night (N), Intersection (I), Merge and Split (M&S). PersFormer\* denotes the latest performance reported on the official code base, Anchor3DLane-T represents the temporal multi-frame method of [12].

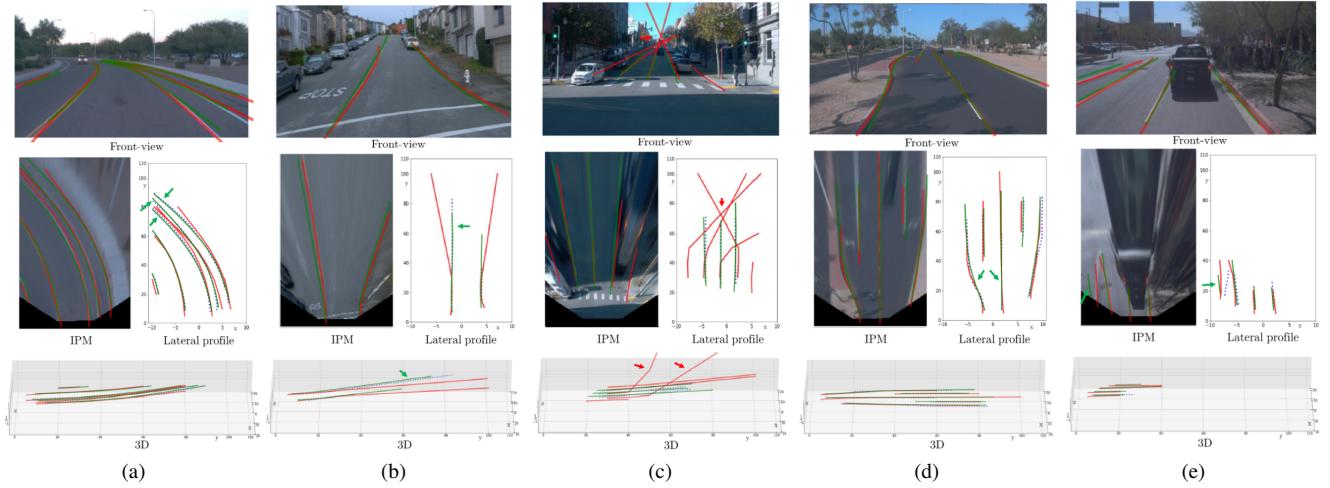


Figure 6. Qualitative comparison on OpenLane. Our method is compared to PersFormer\* with ground truth visualized as dashed lines.

**Implementation details.** We use input size  $360 \times 480$  and adopt the same backbone as in [2] based on a modified EfficientNet [43]. We extract four feature maps of resolutions  $[\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}]$ . The final 3D feature map has size  $26 \times 16$  with 64 channels. We use  $M = 64$  initial line proposals and B-Splines of degree  $d = 3$  and  $K = 10$  control points. We apply Adam optimizer [15] with an initial learning rate of  $2 \times 10^{-4}$  for OpenLane and  $10^{-4}$  for Apollo and a dataset specific step-wise scheduler. We train for 30 epochs on OpenLane and 300 epochs on Apollo with batch size 16. For more details we refer to the supplementary.

## 4.2. Ablation studies

Table 1 indicates the effect of our proposed prior-based regularization. It is evident that each prior improves the F1-Score as well as geometric errors. While the surface and curvature priors result in better far-range estimates, line parallelism supports X-regression in the near-range. Besides, using surface smoothness loss results in lowest Z-far errors. Finally, a combination of priors yields a good balance of

F1-Score and geometric errors. The positive effect of parallelism is confirmed by Fig. 5, where reinforcing parallel lane structure leads to better estimates in the near-range (a) and far-range (b) compared to the unregularized model. Learning parallel lines also is evidently beneficial in cases of poor visibility (b) and occlusions (a). In the latter case, the regularized model even shows better predictions than the noisy ground truth. This emphasizes the high relevance of priors for more robust behavior for real-world datasets, where 3D ground truth often comes with inaccuracies.

For the spatial transformation (see Table 2), too low numbers of surface hypotheses result in worse score, presumably as 3D geometry is not captured sufficiently, whereas larger numbers tend to decreasing performance due to the higher complexity. The best F1-Score is obtained with 5 hypotheses, which is chosen for further experiments. While the improvement over IPM is already considerable, we think that with the simplifications of plane hypotheses prevent the component from developing its full potential. We see ways to enhance the 3D transformation even further

Method	Balanced Scenes				Rare Scenes					
	F1(%)↑	X-error (m) ↓ near	X-error (m) ↓ far	Z-error (m) ↓ near	Z-error (m) ↓ far	F1(%)↑	X-error (m) ↓ near	X-error (m) ↓ far	Z-error (m) ↓ near	Z-error (m) ↓ far
3D-LaneNet [6]	86.4	0.068	0.477	0.015	<b>0.202</b>	72.0	0.166	0.855	0.039	<b>0.521</b>
GP [18]	91.9	0.049	0.387	<b>0.008</b>	0.213	83.7	0.126	0.903	<u>0.023</u>	0.625
PersFormer [2]	92.9	0.054	0.356	0.01	0.234	87.5	0.107	0.782	0.024	0.602
3D-SpLineNet [33]	96.3	0.037	0.324	<u>0.009</u>	0.213	92.9	0.077	0.699	<b>0.021</b>	0.562
CurveFormer [1]	95.8	0.078	0.326	0.018	0.219	95.6	0.182	0.737	0.039	0.561
BEV-LaneDet [47]	96.9	<b>0.016</b>	<b>0.242</b>	0.02	0.216	<b>97.6</b>	<b>0.031</b>	<b>0.594</b>	0.040	0.556
Anchor3DLane [12]	95.4	0.045	0.300	0.016	0.223	94.4	0.082	0.699	0.030	0.580
LaneCPP (Ours)	<b>97.4</b>	<u>0.030</u>	<u>0.277</u>	0.011	<u>0.206</u>	<b>96.2</b>	<u>0.073</u>	<u>0.651</u>	<u>0.023</u>	<u>0.543</u>

Table 5. Quantitative comparison of best methods on Apollo 3D Synthetic [9]. **Best performance** and second best are highlighted.

using more sophisticated spatial representations in future.

The impact of our different contributions is summarized in Table 3, where the first row shows our baseline (see Sec. 4.1). More than two percent in F1-Score are gained with our novel lane representation compared to the simplified one from [33]. Moreover, it is clear that both, the regularization using combined priors and the spatial transformation using 5 hypotheses result in significant improvement. Eventually, combining all components yields the best model configuration, which we choose for further evaluation.

### 4.3. Evaluation on OpenLane

On the real-world OpenLane benchmark our model evidently outperforms all other methods with respect to F1-Score as well as geometric errors as shown in Table 4. Compared to BEV-Lanedet, which achieves a high detection score, our model gains +1.9 %, while reaching significantly lower geometric errors. In comparison to Anchor3DLane the improvements with respect to X-errors are less substantial, however, our approach surpasses the F1-Score by a large gap of +6.6 %. Analyzing the detection scores among different scenarios, outstanding performance gain is observed on the up- and down-hill test set (+5.9 %) that highlights the capability of our approach to capture 3D space proficiently, which is supported by the low Z-errors.

Apart from quantitative results, we show qualitative examples in Fig. 6. In up-hill scenarios like Fig. 6b our model manages to estimate both lateral and height profile accurately, since our assumptions about road surface and line parallelism are satisfied. In contrast, PersFormer lacks spatial features and does not use any kind of physical regularization. Consequently, it fails to estimate the 3D lane geometry and even collapses in Fig. 6c, whereas our surface and curvature priors always prevent such a behavior. Noteworthy is also the top performance on the merges and splits set. This proves that our soft regularization is even capable to handle situations containing non-parallel lines, which is also confirmed by Fig. 6d. However, we rarely observe limi-

tations with our formulation for line pairs with a similar orientation but weakly converging course as shown in Fig. 6e. In such cases the indicator function might erroneously decide for parallelism loss during training. One possible solution for future work would be to consider ground truth for the indicator function to identify such situations.

### 4.4. Evaluation on Apollo 3D Synthetic

The Apollo 3D Synthetic dataset is very limited in size and only consists of simple situations in contrast to OpenLane. While we find the results on OpenLane more meaningful, we would like to still provide and discuss the quantitative results on the Apollo dataset. Due to the simplicity of the dataset, our model cannot benefit that significantly from our priors but still achieves competitive results to state of the art with the highest F1-Score on the balanced scenes dataset and comparable error metrics (second best for most errors).

## 5. Conclusions and future work

In this work, we present LaneCPP, a novel approach for 3D lane detection that leverages physical prior knowledge about lane structure and road geometry. Our new continuous lane representation overcomes previous deficiencies by allowing arbitrary lane structures and enables us to regularize lane geometry based on analytically formulated priors. We further introduce a novel spatial transformation module that models 3D features carefully considering knowledge about road surface geometry. In our experiments, we demonstrate state-of-the-art performance on real and synthetic benchmarks. The full capability of our approach is revealed on real-world OpenLane, for which we prove the relevance of priors quantitatively and qualitatively. In future, priors could be individualized for different driving scenarios and might support to learn inter-lane relations to achieve better scene understanding in a global context. We also see ways to leverage the full potential of the spatial transformation by using more sophisticated surface representations.

## 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. 1, 2, 3, 7, 8
- [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. 2, 3, 4, 6, 7, 8, 9, 10, 11
- [3] Carl de Boor. On calculating with b-splines. *Journal of Approximation Theory*, 1972. 2
- [4] Netalee Efrat, Max Bluvstein, Shaul Oron, Dan Levi, Noa Garnett, and Bat El Shlomo. 3d-lanenet+: Anchor free lane detection using a semi-local representation. *arXiv/2011.01535*, 2020. 1, 2, 3
- [5] Zhengyang Feng, Shaohua Guo, Xin Tan, Ke Xu, Min Wang, and Lizhuang Ma. Rethinking efficient lane detection via curve modeling. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2
- [6] 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. 1, 2, 3, 7, 8
- [7] Mohsen Ghafoorian, Cedric Nugteren, Nóra Baka, Olaf Booij, and Michael Hofmann. EL-GAN: embedding loss driven generative adversarial networks for lane detection. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2018. 1, 2
- [8] Stephen Gould, Richard Hartley, and Dylan Campbell. Deep declarative networks. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2021. 2
- [9] 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. 1, 2, 3, 4, 6, 7, 8, 12
- [10] Steffen Hagedorn, Marcel Milich, and Alexandru P. Condurache. Pioneering se (2)-equivariant trajectory planning for automated driving. *arXiv:2403.11304*, 2024. 2
- [11] Yuenan Hou, Zheng Ma, Chunxiao Liu, and Chen Change Loy. Learning lightweight lane detection cnns by self attention distillation. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2019. 1, 2
- [12] 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. 2, 3, 7, 8
- [13] Brody Huval, Tao Wang, Sameep Tandon, Jeff Kiske, Will Song, Joel Pazhayampallil, Mykhaylo Andriluka, Pranav Rajpurkar, Toki Migimatsu, Royce Cheng-Yue, Fernando A. Mujica, Adam Coates, and Andrew Y. Ng. An empirical evaluation of deep learning on highway driving. *arXiv/1504.01716*, 2015. 1, 2
- [14] 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. 8
- [15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2015. 7, 5
- [16] YeongMin Ko, Jiwon Jun, Donghwuy Ko, and Moongu Jeon. Key points estimation and point instance segmentation approach for lane detection. *arXiv/2002.06604*, 2020. 1, 2
- [17] Seokju Lee, Junsik Kim, Jae Shin Yoon, Seunghak Shin, Oleksandr Bailo, Namil Kim, Tae-Hee Lee, Hyun Seok Hong, Seung-Hoon Han, and In So Kweon. Vpgnet: Vanishing point guided network for lane and road marking detection and recognition. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2017. 1, 2
- [18] 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. 2, 3, 8
- [19] Qi Li, Yue Wang, Yilun Wang, and Hang Zhao. Hdmapnet: An online HD map construction and evaluation framework. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2022. 3
- [20] Xiang Li, Jun Li, Xiaolin Hu, and Jian Yang. Line-cnn: End-to-end traffic line detection with line proposal unit. *IEEE Trans. on Intelligent Transportation Systems (T-ITS)*, 2020. 1, 2
- [21] Zhiqi Li, Wenhui Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Yu Qiao, and Jifeng Dai. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2022. 3
- [22] 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. 6, 4
- [23] Ruijin Liu, Zejian Yuan, Tie Liu, and Zhiliang Xiong. End-to-end lane shape prediction with transformers. In *Proc. of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2021. 2
- [24] 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. 1, 2, 3, 8
- [25] Pingping Lu, Chen Cui, Shaobing Xu, Huei Peng, and Fan Wang. SUPER: A novel lane detection system. *IEEE Trans. on Intelligent Vehicles (T-IV)*, 2021. 2
- [26] Hanspeter Mallot, Heinrich Bülfhoff, J.J. Little, and S Bohrer. Inverse perspective mapping simplifies optical flow computation and obstacle detection. *Biological Cybernetics*, 1991. 3
- [27] Davy Neven, Bert De Brabandere, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Towards end-to-end lane detection: an instance segmentation approach. In *Proc. IEEE Intelligent Vehicles Symposium (IV)*, 2018. 1, 2

- [28] Marcos Nieto, Luis Salgado, Fernando Jaureguizar, and Jon Arrospide. Robust multiple lane road modeling based on perspective analysis. In *Proc. IEEE International Conf. on Image Processing (ICIP)*, 2008. 2
- [29] Bowen Pan, Jiankai Sun, Ho Yin Tiga Leung, Alex Andonian, and Bolei Zhou. Cross-view semantic segmentation for sensing surroundings. *IEEE Robotics Autom. Lett.*, 2020. 3
- [30] Xingang Pan, Jianping Shi, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Spatial as deep: Spatial CNN for traffic scene understanding. In *Proc. of the Conf. on Artificial Intelligence (AAAI)*, 2018. 1, 2
- [31] Lang Peng, Zhirong Chen, Zhangjie Fu, Pengpeng Liang, and Erkang Cheng. Bevsegformer: Bird's eye view semantic segmentation from arbitrary camera rigs. In *Proc. of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2023. 3
- [32] 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. 3, 5, 6
- [33] 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. 1, 2, 3, 4, 6, 8, 5
- [34] Fabio Pizzati, Marco Allodi, Alejandro Barrera, and Fernando García. Lane detection and classification using cascaded cnns. In *Proc. of the International Conf. on Computer Aided Systems Theory (EUROCAST)*, 2019. 1, 2
- [35] Zhan Qu, Huan Jin, Yang Zhou, Zhen Yang, and Wei Zhang. Focus on local: Detecting lane marker from bottom up via key point. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1, 2
- [36] Matthias Rath and Alexandru Paul Condurache. Invariant integration in deep convolutional feature space. In *Proc. of European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, 2020. 2
- [37] Matthias Rath and Alexandru Paul Condurache. Improving the sample-complexity of deep classification networks with invariant integration. In *Proc. of International Joint Conf. on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP)*, 2022. 2
- [38] Thomas Roddick and Roberto Cipolla. Predicting semantic map representations from images using pyramid occupancy networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3
- [39] Thomas Roddick, Alex Kendall, and Roberto Cipolla. Orthographic feature transform for monocular 3d object detection. In *Proc. of the British Machine Vision Conf. (BMVC)*, 2019. 3
- [40] Avishkar Saha, Oscar Mendez, Chris Russell, and Richard Bowden. Translating images into maps. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2022. 3
- [41] Jinming Su, Chao Chen, Ke Zhang, Junfeng Luo, Xiaoming Wei, and Xiaolin Wei. Structure guided lane detection. In *Proc. of the International Joint Conf. on Artificial Intelligence (IJCAI)*, 2021. 2
- [42] Lucas Tabelini, Rodrigo Berriel, Thiago M Paixao, Claudine Badue, Alberto F De Souza, and Thiago Oliveira-Santos. Keep your eyes on the lane: Real-time attention-guided lane detection. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1, 2
- [43] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proc. of the International Conf. on Machine learning (ICML)*, 2019. 7, 2
- [44] Lucas Tabelini Torres, Rodrigo Ferreira Berriel, Thiago M. Paixão, Claudine Badue, Alberto F. De Souza, and Thiago Oliveira-Santos. Polylanenet: Lane estimation via deep polynomial regression. In *Proc. of the International Conf. on Pattern Recognition (ICPR)*, 2020. 2
- [45] Bingke Wang, Zilei Wang, and Yixin Zhang. Polynomial regression network for variable-number lane detection. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020. 2
- [46] Jinsheng Wang, Yinchao Ma, Shaofei Huang, Tianrui Hui, Fei Wang, Chen Qian, and Tianzhu Zhang. A keypoint-based global association network for lane detection. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 2
- [47] 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. 2, 3, 7, 8
- [48] Yuping Wang and Jier Chen. Eqdrive: Efficient equivariant motion forecasting with multi-modality for autonomous driving. *arXiv:2310.17540*, 2023. 2
- [49] Lu Xiong, Zhenwen Deng, Peizhi Zhang, and Zhiqiang Fu. A 3d estimation of structural road surface based on lane-line information. *IFAC Conf. on Engine and Powertrain Control, Simulation and Modeling (E-COSM)*, 2018. 2
- [50] Chenxin Xu, Robby T. Tan, Yuhong Tan, Siheng Chen, Yu Guang Wang, Xinchao Wang, and Yanfeng Wang. Eqmotion: Equivariant multi-agent motion prediction with invariant interaction reasoning. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2
- [51] Fan Yan, Ming Nie, Xinyue Cai, Jianhua Han, Hang Xu, Zhen Yang, Chaoqiang Ye, Yanwei Fu, Michael Bi Mi, and Li Zhang. Once-3dlanes: Building monocular 3d lane detection. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2, 3
- [52] Tu Zheng, Hao Fang, Yi Zhang, Wenjian Tang, Zheng Yang, Haifeng Liu, and Deng Cai. RESA: recurrent feature-shift aggregator for lane detection. In *Proc. of the Conf. on Artificial Intelligence (AAAI)*, 2021. 2
- [53] Tu Zheng, Yifei Huang, Yang Liu, Wenjian Tang, Zheng Yang, Deng Cai, and Xiaofei He. Clrnet: Cross layer refinement network for lane detection. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2
- [54] Qin Zou, Hanwen Jiang, Qiyu Dai, Yuanhao Yue, Long Chen, and Qian Wang. Robust lane detection from continuous driving scenes using deep neural networks. *IEEE Trans. on Vehicular Technology (VTC)*, 2020. 1, 2

## Supplementary Material

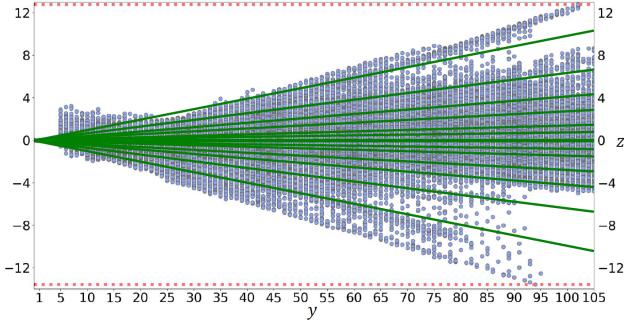


Figure 7. Height distribution ( $z$ ) along the longitudinal direction ( $y$ ) of ground truth line points (blue points) on OpenLane dataset. Height deviations in the near-range (left side) tend to be smaller than in the far-range (right side) spanning a triangle-like region of interest in the  $y$ - $z$ -profile. For the spatial transformation, we sample surface hypotheses (green) of different pitch angles to cover this region.

## A. Architecture Details

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

### A.1. Backbone

Similar to [2], we use a modified version of EfficientNet [43] 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  $180 \times 240$ ,  $90 \times 120$ ,  $45 \times 60$ ,  $22 \times 30$ . 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 M.

### A.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  $1 \times 1$  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  $22 \times 30$  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	Pitch Angles
1	$\{0^\circ\}$
3	$\{-2^\circ, 0^\circ, 2^\circ\}$
5	$\{-2^\circ, -1^\circ, 0^\circ, 1^\circ, 2^\circ\}$
15	$\{-5^\circ, -2^\circ, -1.7^\circ, -1.3^\circ, -1^\circ, -0.7^\circ, -0.3^\circ, 0^\circ, 0.3^\circ, 0.7^\circ, 1^\circ, 1.3^\circ, 1.7^\circ, 2^\circ, 5^\circ\}$
27	$\{-10^\circ, -8.5^\circ, -7^\circ, -5.8^\circ, -4.5^\circ, -3.3^\circ, -2^\circ, -1.7^\circ, -1.4^\circ, -1^\circ, -0.8^\circ, -0.6^\circ, -0.3^\circ, 0^\circ, 0.3^\circ, 0.6^\circ, 0.8^\circ, 1^\circ, 1.4^\circ, 1.7^\circ, 2^\circ, 3.3^\circ, 4.5^\circ, 5.8^\circ, 7^\circ, 8.5^\circ, 10^\circ\}$

Table 6. Different orientations of surface hypotheses.

surface hypotheses such that the distribution of lane height is covered (see Fig. 7). 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 6.

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  $208 \times 128$ ,  $104 \times 64$ ,  $52 \times 32$ ,  $26 \times 16$ .

### A.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  $26 \times 16$ . 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  $26 \times 16 \times 64$ .

### A.4. Detection head

The detection head operates on a BEV feature map of shape  $26 \times 16 \times 64$  covering a range of  $[-10 \text{ m}, 10 \text{ m}]$  in lateral  $x$ -direction and  $[3 \text{ m}, 103 \text{ m}]$  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. 8. 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  $\text{max\_cells}$ . We then take the 64-dimensional features of the set of selected cells and flatten it to a feature vector of size  $64 \cdot \text{max\_cells}$ . If less than  $\text{max\_cells}$  are pooled for the proposal, the remaining entries 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. 8. Important to notice is also that the fully connected layers share

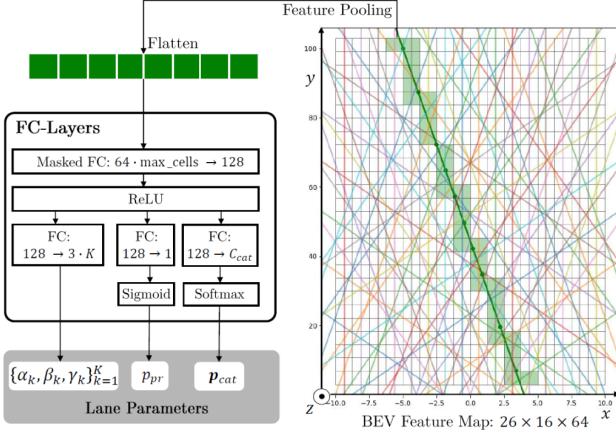


Figure 8. The detection head of our model: First, features are pooled from the BEV feature map for each proposal. Afterwards, pooled features are flattened and fed through several fully-connected (FC) layers, which share weights for all proposals, to finally obtain the lane parameters.

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  $\mathbf{p}_{cat}$  for different line categories.

## B. Training

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

### B.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. 9.

The matching of ground truth lines to the line proposals is inspired by [33], 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  $\bar{\mathbf{f}}$  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

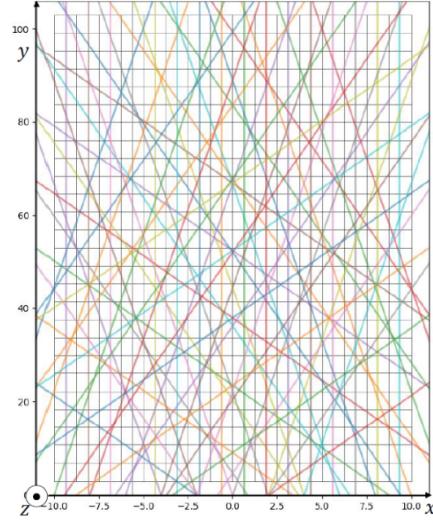


Figure 9. Visualization of different initial line proposals. Colorful lines represent the line proposals. The black lines show the grid of the final BEV feature map.

given as

$$L^{(ij)} = \lambda_{UCD} \cdot UCD(\bar{\mathbf{f}}^{(i)}, \mathcal{P}_{GT}^{(j)}) + \quad (14)$$

$$\lambda_{CosD} \cdot CosD(\bar{\mathbf{f}}^{(i)}, \mathcal{P}_{GT}^{(j)}), \quad (15)$$

with weights for each cost component  $\lambda_{UCD}$  and  $\lambda_{CosD}$ . 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  $L^{(ij)} < L_{thr}$ .

### B.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  $\mathbb{1}_p^{(ij)}$  deciding, whether the loss is applied to the point pair consisting of point  $p$  on line  $i$  and the best matching point in normal direction  $p^*$  on line  $j$ . The indicator function is defined as

$$\mathbb{1}_p^{(ij)} = \begin{cases} 1 & \text{if } OD_{p^*}^{(ij)} < OD_{thr} \text{ and } \sigma^{(ij)} < \sigma_{thr}, \\ 0 & \text{else.} \end{cases} \quad (16)$$

As Eq. (16) shows, the parallelism criterion holds if two conditions are fulfilled. The first condition  $OD_{p^*}^{(ij)} < OD_{thr}$  takes into account the orthogonal distance ( $OD$ ) of the best matching point  $p^*$  on line  $j$  to the normal plane spanned by the tangent  $\mathbf{T}^{(i)}(t_p)$  at point  $p$  on line  $i$ , which is given as

$$OD_{p^*}^{(ij)} = \mathbf{T}^{(i)}(t_p)^T \cdot (\mathbf{f}^{(j)}(t_{p^*}) - \mathbf{f}^{(i)}(t_p)). \quad (17)$$

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  $D_p^{(ij)}$  of point pairs of neighboring lines  $i$  and  $j$ . The standard deviation is defined as

$$\sigma^{(ij)} = \sqrt{\frac{1}{|\mathcal{P}^{(i)}|} \sum_{p \in \mathcal{P}^{(i)}} D_p^{(ij)} - \bar{D}^{(ij)}}, \text{ where} \quad (18)$$

$$\bar{D}^{(ij)} = \frac{1}{|\mathcal{P}^{(i)}|} \sum_{p \in \mathcal{P}^{(i)}} D_p^{(ij)}, \quad (19)$$

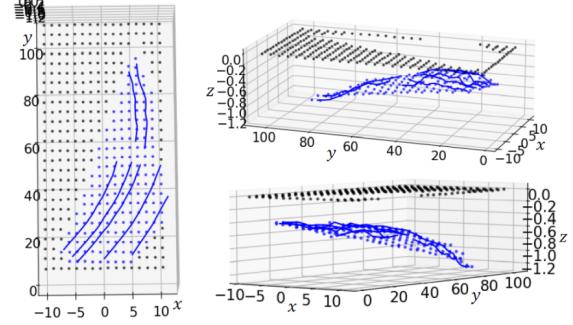
and the euclidean distance for one point pair as  $D_p^{(ij)} = \|\mathbf{f}^{(i)}(t_p) - \mathbf{f}^{(j)}(t_{p^*})\|_2$ . For lines of different orientations (as for merging and splitting lines) this standard deviation is rather high and more likely surpasses the threshold  $\sigma_{thr}$  in contrast 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. 10 for an up-hill and a down-hill scenario.

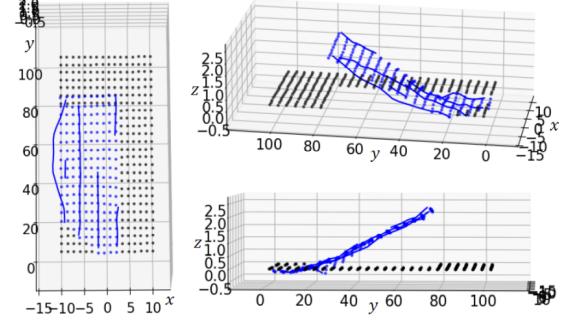
**Lane presence and category classification losses.** For both classification losses, we apply focal loss [22]. For line presence, which only considers the two classes present and not present, the loss is given as

$$\mathcal{L}_{pr} = -\frac{1}{M} \sum_{i=1}^M \left( \hat{p}_{pr}^{(i)} \cdot (1 - p_{pr}^{(i)})^{\gamma_f} \cdot \log(p_{pr}^{(i)}) + \right. \quad (20)$$

$$\left. (1 - \hat{p}_{pr}^{(i)}) \cdot (p_{pr}^{(i)})^{\gamma_f} \cdot \log(1 - p_{pr}^{(i)}) \right), \quad (21)$$



(a) Down-hill scenario



(b) Up-hill scenario

Figure 10. 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.

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 [22] to handle class imbalance.

The category classification loss is applied for datasets, which provide lane category information in the ground truth. Analogously to Eq. (21), the loss is given as

$$\mathcal{L}_{cat} = -\frac{1}{M} \sum_{i=1}^M \frac{1}{C_{cat}} \sum_{c=1}^{C_{cat}} \left( \hat{p}_{cat}^{(i)}[c] \cdot \right. \quad (22)$$

$$\left. (1 - p_{cat}^{(i)}[c])^{\gamma_f} \cdot \log(p_{cat}^{(i)}[c]) \right), \quad (23)$$

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

**Regression loss.** For both, the regression and visibility loss, the curve argument  $t_p$  has to be determined for a respective point in the ground truth  $\mathbf{p} \in \mathcal{P}_{GT}$ . 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. 11. After having obtained the

curve arguments in orthogonal direction, the regression loss for a line proposal  $i$  is given as

$$\mathcal{L}_{reg}^{(i)} = \frac{1}{|\mathcal{P}_{GT}^{(i)}|} \sum_{\mathbf{p} \in \mathcal{P}_{GT}^{(i)}} \hat{v}_{\mathbf{p}}^{(i)} \cdot \left\| \mathbf{w} \odot \left( \mathbf{f}^{(i)}(\mathbf{t}_{\mathbf{p}}) - \begin{pmatrix} \hat{x}_{\mathbf{p}}^{(i)} \\ \hat{y}_{\mathbf{p}}^{(i)} \\ \hat{z}_{\mathbf{p}}^{(i)} \end{pmatrix} \right) \right\|_1 \quad (24)$$

with  $\hat{v}_{\mathbf{p}}^{(i)}$  the ground truth visibility information and  $(\hat{x}_{\mathbf{p}}^{(i)}, \hat{y}_{\mathbf{p}}^{(i)}, \hat{z}_{\mathbf{p}}^{(i)})^T$  the 3D position of a ground truth point  $\mathbf{p}$  on line  $i$ .  $\mathbf{w} \in \mathbb{R}^3$  is a vector with weighting factors for each 3D component providing for a more balanced regression in each dimension. As shown in Eq. (24) and illustrated in Fig. 11a, only visible points are utilized. The total regression loss for all lines is given as

$$\mathcal{L}_{reg} = \frac{1}{M} \sum_{i=1}^M \hat{p}_{pr}^{(i)} \cdot \mathcal{L}_{reg}^{(i)}. \quad (25)$$

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

$$\mathcal{L}_{vis}^{(i)} = -\frac{1}{|\mathcal{P}_{GT}^{(i)}|} \sum_{\mathbf{p} \in \mathcal{P}_{GT}^{(i)}} \hat{v}_{\mathbf{p}}^{(i)} \cdot \log (\sigma(v^{(i)}(\mathbf{t}_{\mathbf{p}}))) + \quad (26)$$

$$(1 - \hat{v}_{\mathbf{p}}^{(i)}) \cdot \log (1 - \sigma(v^{(i)}(\mathbf{t}_{\mathbf{p}}))). \quad (27)$$

As illustrated in Fig. 11b, all points from the ground truth line are considered. The total visibility loss is then given as

$$\mathcal{L}_{vis} = \frac{1}{M} \sum_{i=1}^M \hat{p}_{pr}^{(i)} \cdot \mathcal{L}_{vis}^{(i)}. \quad (28)$$

## C. Additional implementation details

In the following, we provide more implementation details.

### C.1. Matching

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

### C.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  $\mathbf{w} = (2, 10, 1)^T$ . The thresholds for the indicator function used for the prior losses are  $\sigma_{thr} = 2$  m and  $OD_{thr} = 1$  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.

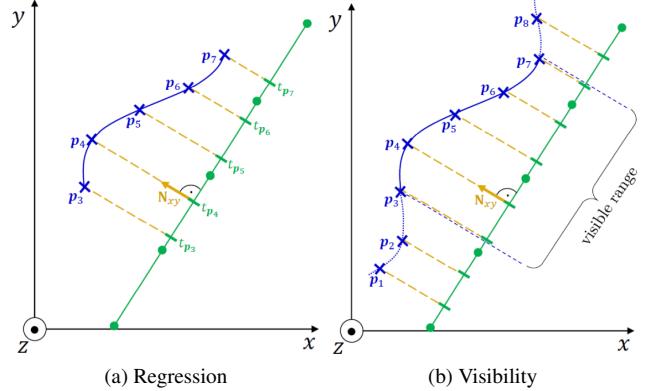


Figure 11. Projection of ground truth points  $\mathbf{p}$  onto line proposal in normal direction to obtain curve arguments  $\mathbf{t}_{\mathbf{p}}$ . For regression (a) only visible points are considered (continuous lines), for visibility (b) all points are taken into account, where invisible points are marked with dashed lines.

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

Table 7. 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).

## C.3. Training procedure

In the training, we use Adam optimizer [15], 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.

## C.4. Others

The maximum number of cells used for feature pooling in the detection head is  $\text{max\_cells} = 64$ .

## D. Additional results

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

### D.1. Ablation studies

Table 7 shows the performance of 3D-SpLineNet [33] 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.

In Table 8 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

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

Table 8. 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.

cloud in 3D space as described in Sec. 3.3 in the main paper. For the uniform sampling (comparable to [32]), the samples are drawn along the rays with equal step size in the range [3 m, 110 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 8, 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.

## D.2. Quantitative results

In Table 9 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 10.

## D.3. Qualitative results

We show additional qualitative results on OpenLane in Fig. 12. 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. 12a - Fig. 12b) and up- or down-hill scenarios (Fig. 12d - Fig. 12f) our model shows high-quality detections compared to PersFormer. For the intersection scenario (Fig. 12c) 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.

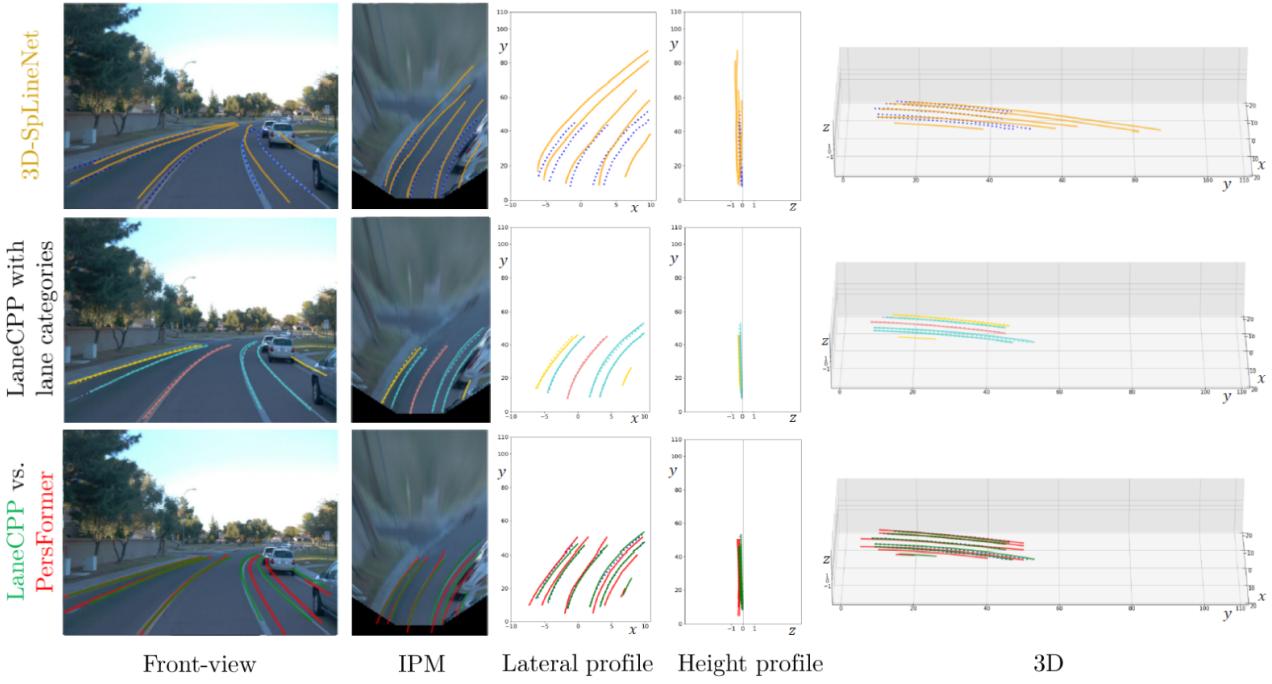
We further demonstrate the results of our model on Apollo 3D Synthetic illustrated in Fig. 13. As shown, our model achieves accurate detection results in simple scenarios from the Balanced Scenes test set (Fig. 13a - Fig. 13b), in more challenging up- and down-hill scenarios from the Rare Scenes test set (Fig. 13c - Fig. 13d) as well as in case of visual variations (Fig. 13e - Fig. 13f). A very challenging scene is shown in Fig. 13f, where our model manages to capture the overall line structure well but still could be improved slightly with respect to close-range  $x$ -errors.

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

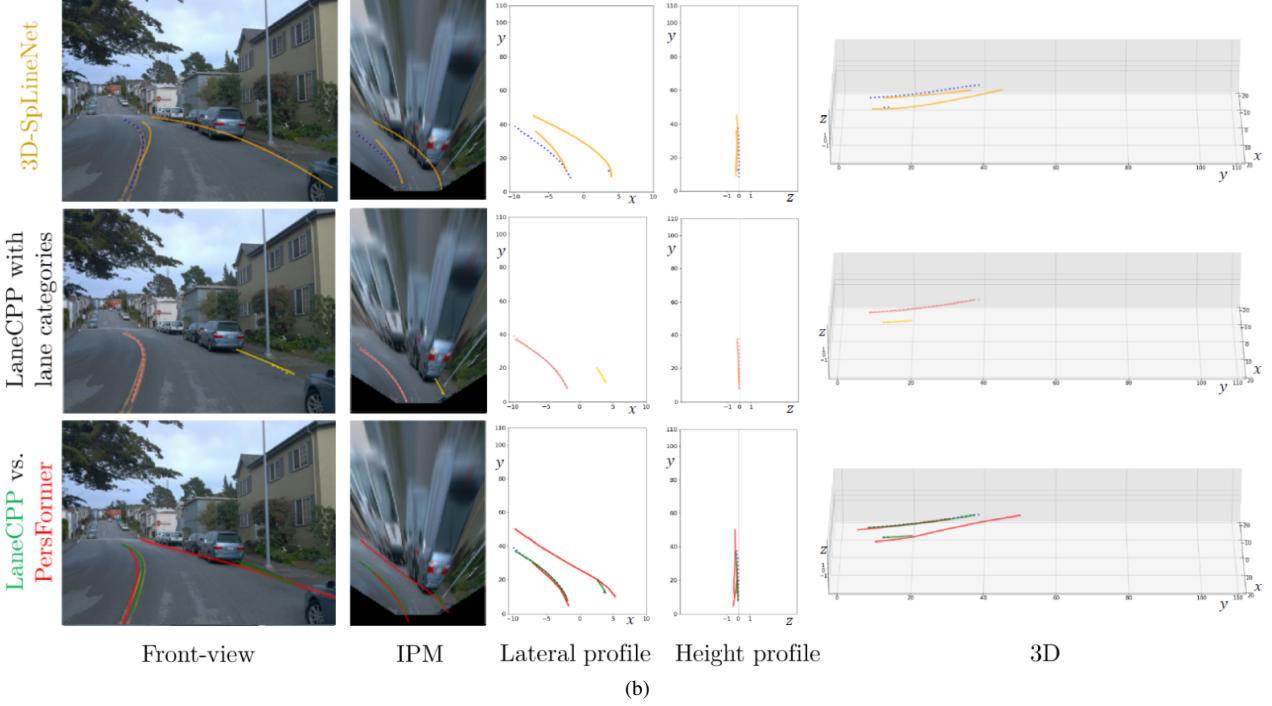
Table 9. Detailed quantitative evaluation of our LaneCPP for different scenarios on OpenLane [2].

Scenario	Method	F1-Score(%)↑	AP(%)↑	X-error (m)↓		Z-error (m)↓	
				near	far	near	far
<i>Balanced Scenes</i>	3D-LaneNet [6]	86.4	89.3	0.068	0.477	0.015	<b>0.202</b>
	Gen-LaneNet [9]	88.1	90.1	0.061	0.496	0.012	0.214
	3D-LaneNet (1/att) [14]	91.0	93.2	0.082	0.439	0.011	0.242
	Gen-LaneNet (1/att) [14]	90.3	92.4	0.08	0.473	0.011	0.247
	CLGO [24]	91.9	94.2	0.061	0.361	0.029	0.250
	GP [18]	91.9	93.8	0.049	0.387	<b>0.008</b>	0.213
	PersFormer [2]	92.9	—	0.054	0.356	0.010	0.234
	3D-SpLineNet [33]	96.3	<u>98.1</u>	0.037	0.324	<u>0.009</u>	0.213
	CurveFormer [1]	95.8	97.3	0.078	0.326	0.018	0.219
	BEV-LaneDet [47]	<u>96.9</u>	—	<b>0.016</b>	<b>0.242</b>	0.02	0.216
<i>Rare Scenes</i>	Anchor3DLane [12]	95.4	97.1	0.045	0.300	0.016	0.223
	LaneCPP	<b>97.4</b>	<b>99.5</b>	<u>0.030</u>	<u>0.277</u>	0.011	<u>0.206</u>
	3D-LaneNet [6]	72.0	74.6	0.166	0.855	0.039	<b>0.521</b>
	Gen-LaneNet [9]	78.0	79.0	0.139	0.903	0.030	0.539
	3D-LaneNet (1/att) [14]	84.1	85.8	0.289	0.925	0.025	0.625
	Gen-LaneNet (1/att) [14]	81.7	83.2	0.283	0.915	0.028	0.653
	CLGO [24]	86.1	88.3	0.147	0.735	0.071	0.609
	GP [18]	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 [33]	92.9	94.8	0.077	0.699	<b>0.021</b>	0.562
<i>Visual Variations</i>	CurveFormer [1]	95.6	<u>97.1</u>	0.182	0.737	0.039	0.561
	BEV-LaneDet [47]	<b>97.6</b>	—	<b>0.031</b>	<b>0.594</b>	0.040	0.556
	Anchor3DLane [12]	94.4	95.9	0.082	0.699	0.030	0.580
	LaneCPP	<u>96.2</u>	<b>98.6</b>	<u>0.073</u>	<u>0.651</u>	<u>0.023</u>	<u>0.543</u>
	3D-LaneNet [6]	72.5	74.9	0.115	0.601	0.032	0.230
	Gen-LaneNet [9]	85.3	87.2	0.074	0.538	0.015	0.232
	3D-LaneNet (1/att) [14]	85.4	87.4	0.118	0.559	0.018	0.290
	Gen-LaneNet (1/att) [14]	86.8	88.5	0.104	0.544	0.016	0.294
	CLGO [24]	87.3	89.2	0.084	0.464	0.045	0.312
	GP [18]	89.9	92.1	0.060	0.446	<b>0.011</b>	0.235
<i>Visual Variations</i>	PersFormer [2]	89.6	—	0.074	0.430	0.015	0.266
	3D-SpLineNet [33]	91.3	<u>93.1</u>	0.069	0.468	<u>0.013</u>	0.248
	CurveFormer [1]	90.8	93.0	0.125	0.410	0.028	0.254
	BEV-LaneDet [47]	<b>95.0</b>	—	<b>0.027</b>	<b>0.320</b>	0.031	0.256
	Anchor3DLane [12]	<u>91.8</u>	92.5	<u>0.047</u>	<u>0.327</u>	0.019	<b>0.219</b>
	LaneCPP	90.4	<b>93.7</b>	0.054	<u>0.327</u>	0.020	0.222

Table 10. Quantitative evaluation on Apollo 3D Synthetic [9]. **Best performance** and second best are highlighted.

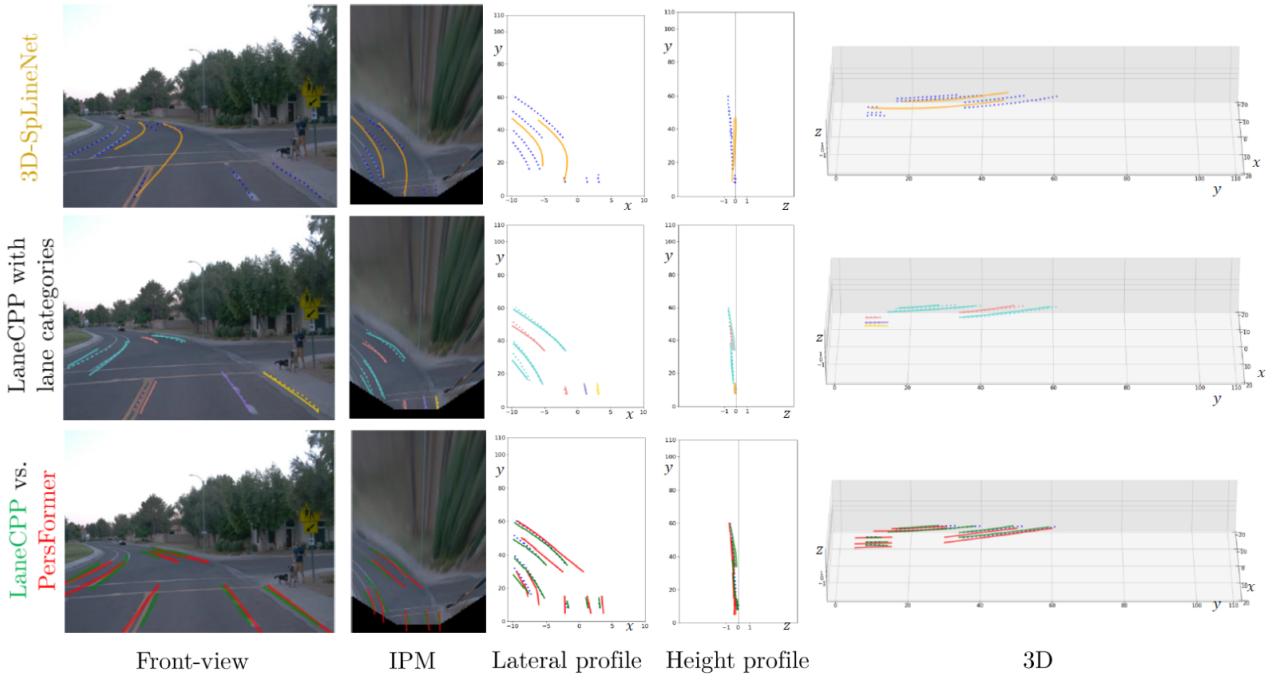


(a)

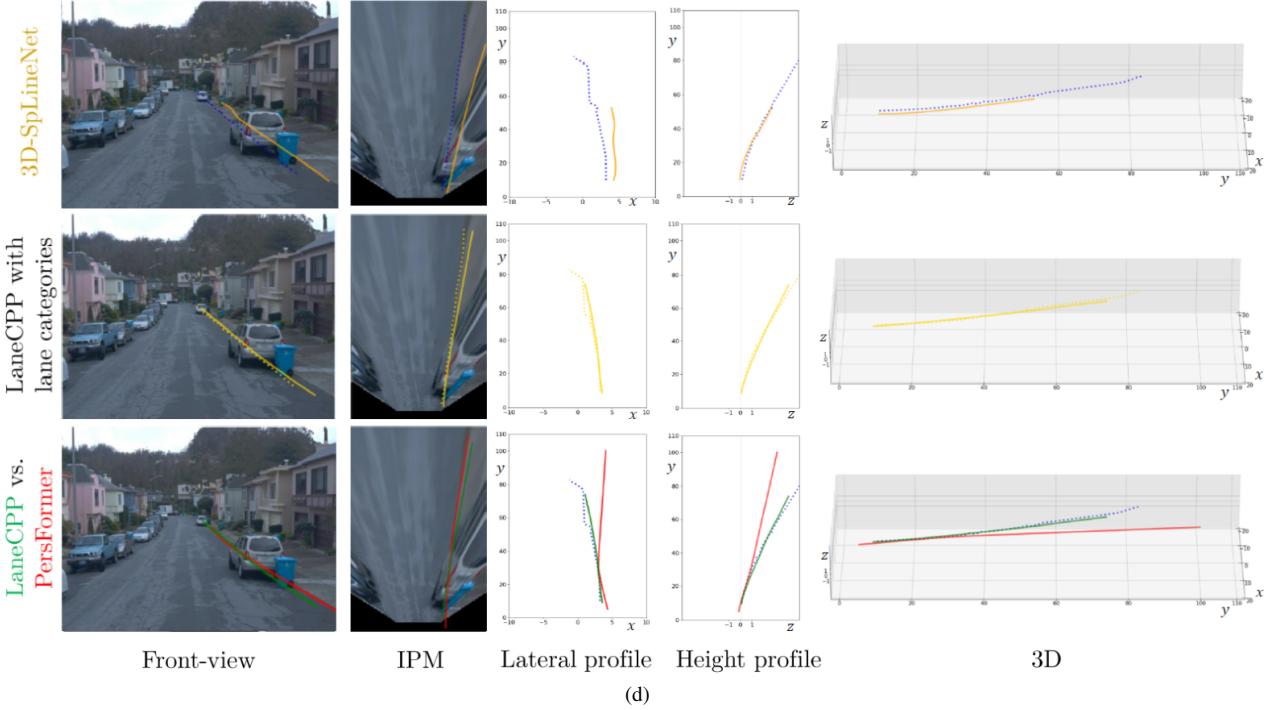


(b)

Figure 12. 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 12. 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\***.

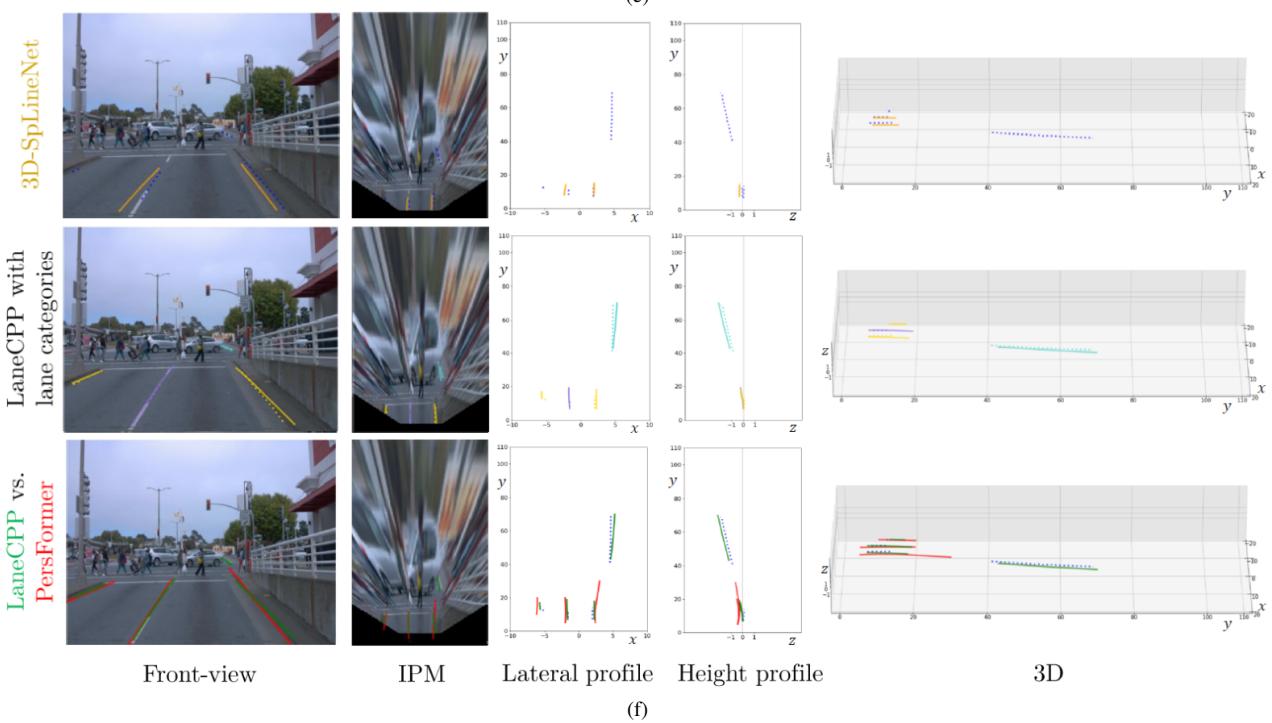
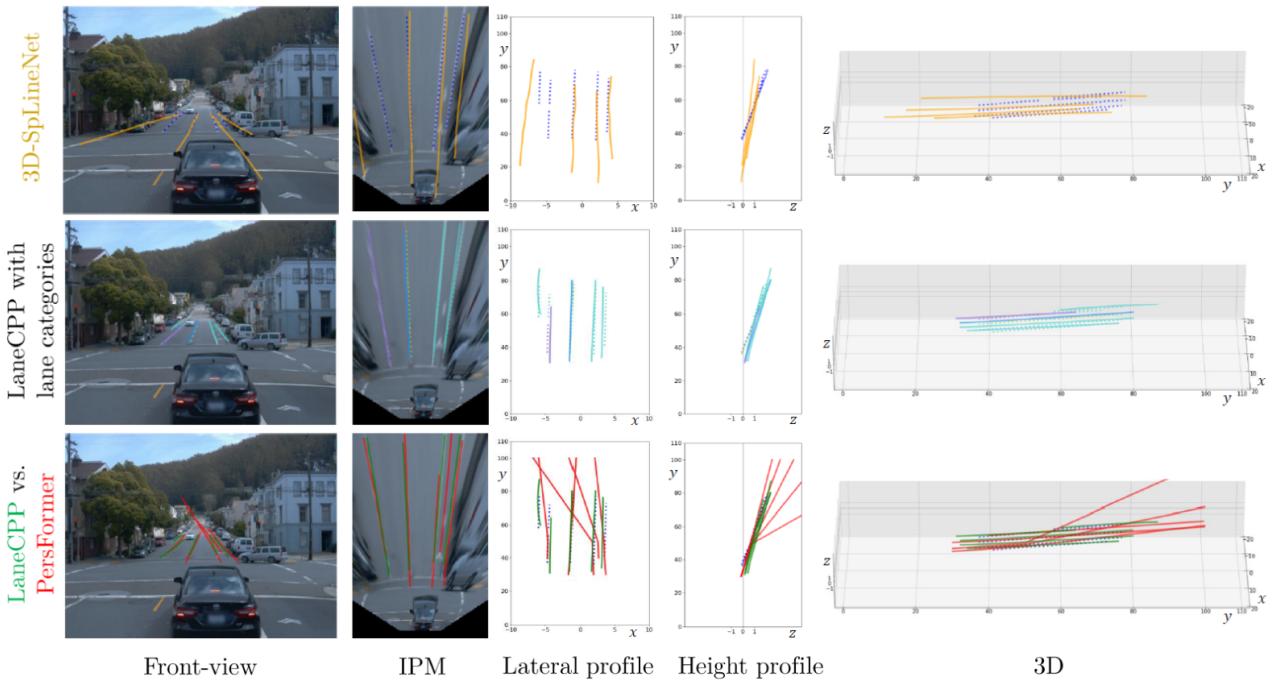


Figure 12. Additional qualitative evaluation on OpenLane [2] test set (3/3). Top row shows **3D-SpLineNet** baseline compared to **ground truth**. Middle row shows LaneCPP with different lane categories illustrated in different colors and ground truth in dashed lines. Bottom row shows direct comparison of **LaneCPP** and **PersFormer\***.

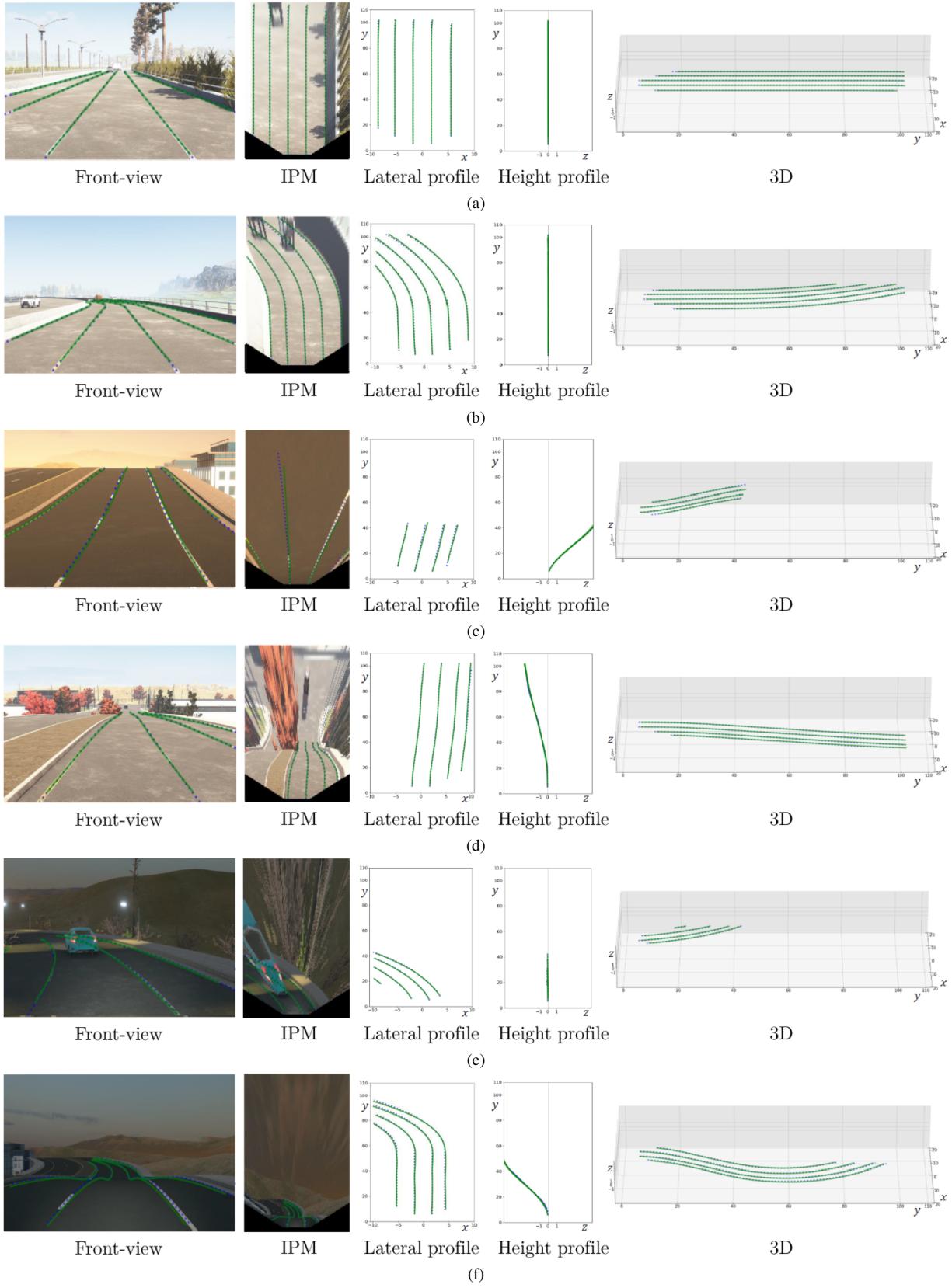


Figure 13. Qualitative evaluation on Apollo 3D Synthetic [9]. Our method is compared to the [ground truth](#) visualized dashed.