3D-LaneNet: end-to-end 3D multiple lane detection

Noa Garnett Rafi Cohen Tomer Pe'er Roee Lahav Dan Levi
General Motors R&D, Israel
{noa.garnett,rafi.cohen,tomer.peer,roee.lahav,dan.levi}@gm.com

Abstract

We introduce a network that directly predicts the 3D layout of lanes in a road scene from a single image. This work marks a first attempt to address this task with on-board sensing instead of relying on pre-mapped environments. Our network architecture, 3D-LaneNet, applies two new concepts: intra-network inverse-perspective mapping (IPM) and anchor-based lane representation. The intra-network IPM projection facilitates a dual-representation information flow in both regular image-view and top-view. An anchor-per-column output representation enables our end-to-end approach replacing common heuristics such as clustering and outlier rejection. In addition, our approach explicitly handles complex situations such as lane merges and splits. Promising results are shown on a new 3D lane synthetic dataset. For comparison with existing methods, we verify our approach on the image-only tuSimple lane detection benchmark and reach competitive performance.

1. Introduction

3D lane detection, comprising of an accurate estimation of the 3D position of the drivable lanes relative to the host vehicle, is a crucial enabler for autonomous driving. Two complementary technological solutions exist: loading pre-mapped lanes generated off-line [31] and perception-based real-time lane detection [4]. The off-line solution is geometrically accurate given precise host localization (in map coordinates) but complex to deploy and maintain. The most common perception-based solution uses a monocular camera as the primary sensor for solving the task. Existing camera-based methods detect lanes in the image domain and then project them to the 3D world by assuming a flat ground [4], leading to inaccurate estimation when the assumption is violated. Inspired by recent success of convolutional neural networks (CNNs) in monocular depth estimation [19], we propose instead to directly detect lanes in 3D. More formally, given a single image taken from a front-facing camera, the task is to output a set of lane entities, each represented as a 3D curve in camera coordinates and describing either a lane delimiter or a lane centerline.

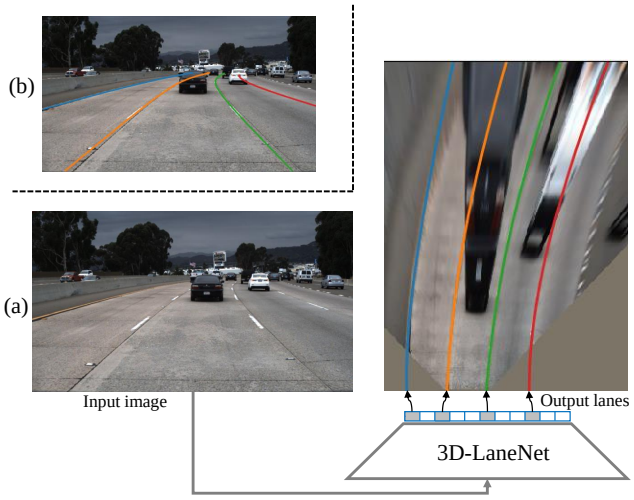

Figure 1. (a) Schematic illustration of the end-to-end approach and lane detection result example on top-view. (b) Projection of the result on the original image

We present 3D-LaneNet, a deep CNN that performs 3D lane detection. The network, trained end-to-end, outputs in each longitudinal road slice, the confidence that a lane passes through the slice and its 3D curve in camera coordinates. Our approach is schematically illustrated in Figure 1(a). Our direct, single-shot approach avoids post-processing used in existing methods such as clustering and outlier rejection. The network's backbone is based on a novel dual pathway backbone architecture that uses several in-network projections of feature maps to virtual bird-eye-view. This dual representation gives the network an enhanced ability to infer 3D in a road scene. The output is represented by a new column-based anchor encoding which makes the network horizontally invariant and enables the end-to-end approach. Each output is associated to an anchor in analogy to single-shot, anchor-based object detection methods such as SSD [20] and YOLO [27].

Acquiring ground truth labeled data with 3D for the task is an endeavor that requires a complex multiple-sensor setup and expensive HD maps. In this work we train and test our

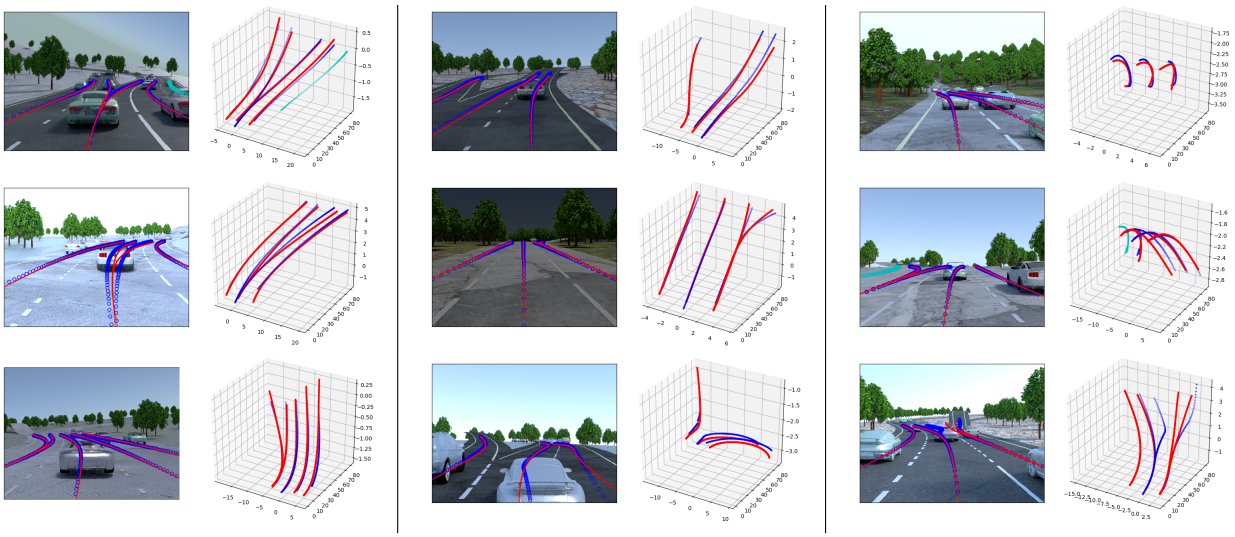

Figure 2. **Examples of 3D lane centerline estimation results (with confidence > 0.5) on test images from the synthetic-3D-lanes dataset.** Ground truth *(blue)* and method result *(red)* shown in each image alongside a 3D visualization. Note that 3D axes scale varies according to scene. Ignored lanes are marked in *cyan*. The bottom rightmost example shows a failure probably due to occlusion in which the split is mistakenly attributed to the right lane.

network using a new synthetic (computer graphics) dataset, *synthetic-3D-lanes* [1] providing full access to the exact 3D position of each lane element. The dataset is currently limited to highway scenarios. While several driving simulators exist [7, 28], they are not focused on the 3D lane detection task, and are limited in the variability of the relevant scene properties (e.g. lane curvature). Our main accomplishment in this domain, is the ability to randomly generate highway segments with variable 3D shapes and lane topology. While the main strength of our approach is in estimating 3D lanes, it can also be used for image-only lane detection. We trained and tested our *direct* method on the tuSimple dataset [1] and reached results competitive to state-of-the-art without the common post-processing techniques.

To summarize, our main contributions are:

- Definition and evaluation metrics for the 3D lane detection task, as well as a first proposed solution
- A novel dual-pathway architecture with intra-network feature map IPM projections
- An new anchor-based output representation of lanes enabling a direct, end-to-end trained network, for both 3D and image-based lane detection.
- A methodology for generating randomly synthetic examples with variation in lane topology (i.e. number of lanes, merges, splits) and 3D shape.

[1] will be uploaded

2. Related Work

Traditional lane detection systems (e.g. [9]) combine low-level operations such as directional filters with high-level heuristics such as the hough transform to detect continuous lanes in the image. A common pipeline includes 4 stages: local lane-feature extraction (1), lane model fitting (2), image-to-world correspondence (3) and temporal aggregation (4). Bar-Hillel *et al.* [4] provide this modular decomposition alongside a detailed overview of traditional systems. In recent years, the *local feature extraction stage* is performed by applying one or more CNNs to the image, but the overall pipeline remains very similar and the latter post processing stages remain.

Initially, CNNs were used to improve feature extraction by either enhancing the edge map (Kim and Lee [14]) or classifying candidate patches (He *et al.* [11]). Huval *et al.* [12] detects local lane line segments with an object detection CNN. VPGNet (Lee *et al.* [17] follows a similar concept and additionally detects other road markings and the vanishing point to improve lane detection. Kim and Park [15] re-formulate the local-feature extraction stage as a semantic-segmentation problem, with two classes corresponding to the left and right lane delimiters, extending the reach of the network to perform clustering. However, a world-coordinate lane model must still be fitted to each cluster, and multiple lanes are not handled. Neven *et al.* [23] make an attempt towards end-to-end multi-lane detection, by training a CNN not only to create a binary lane pixel mask but also a feature embedding used for clustering lane points. Ghafoorian *et al.* [8] propose applying a generative

adversarial network to make the semantic segmentation network output more realistic in the context of lane detection. Several works (e.g. Meyer *et al.* [22], Oliveira *et al.* [24]) are built on a similar approach in which the host and possibly adjacent lanes are the semantic classes (lane interior rather than the lane delimiters).

As opposed to all presented methods, 3D-LaneNet unifies the first three stages of the common pipeline by providing a full multi-lane representation in 3D world coordinates directly from the image in a single feed-forward pass. In addition, previous methods use the flat ground assumption for the image-to-world correspondence while our method estimates the full 3D shape of the lanes.

Inverse perspective mapping (IPM) generates a virtual top-view (sometimes called bird-eye-view) of the scene from a camera-view as in the example in Figure 1. It was introduced in the context of obstacle detection by Mallot *et al.* [21] and first used for lane detection by Pomerleau [26]. IPM has since been extensively used for lane detection (e.g. [5, 3]) since lanes are ordinarily parallel in this view and their curvature can be accurately fitted with low-order polynomials. In addition, removing the perspective effects causes lane markings to look similar (except for blurring effects) regardless of their distance from the camera. Most recently He *et al.* [11] introduced a "Dual-view CNN" which is composed of two separate sub-networks, each producing a descriptor (one per view) which are then concatenated and applied to candidate image locations. Li *et al.* [18] use a CNN to detect lane markings along with geometrical attributes, such as local position and orientation, directly on a top-view image which preserves invariance to these properties. In addition they deploy a second, recurrent network, that traverses the image to detect consistent lanes. Neven *et al.* [23] use the horizon, predicted in each image by a sub-network ("H-net"), to project the lanes to top-view for improved curve fitting. In contrast to previous work, we exploit both views in a synergistic single network approach.

More generally, we propose the first method that uses and end-to-end trained CNN to directly detect multiple lanes and estimate the 3D curvature for each such lane. We also show that our method is applicable both to centerlines and delimiters with an ability to handle splits and merges as well, without any further post-processing.

3. Method

Our method gets as input a single image taken from a front facing camera mounted on a vehicle as illustrated in Figure 4. We assume known intrinsic camera parameters κ (e.g. focal length, center of projection). We also assume that the camera is installed at zero degrees roll *relative to the local ground plane*. We do not assume a known height and pitch since these may change due to vehicle dynamics. The lanes in a road scene can be described both by the set

Figure 3. **Annotated example.** Lane centerlines are marked in blue and delimiters in yellow dotted curves.

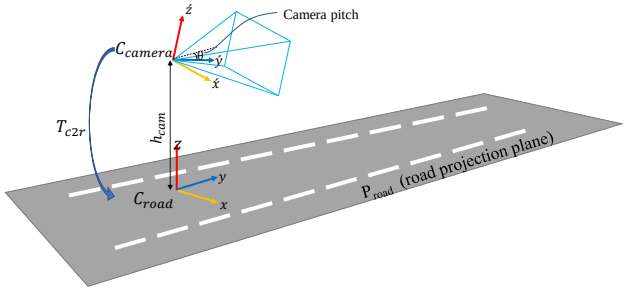

Figure 4. **Camera position and road projection plane**

of centerlines $\{C_i\}_{i=1}^{N_C}$ of each lane and by the set of lane delimiters $\{D_i\}_{i=1}^{N_D}$ as shown in Fig. 3. Each such lane entity (centerline or delimiter) is a curve in 3D expressed in camera coordinates ($\mathcal{C}_{camera}$). We define the task as detecting either the set of lane centerlines and/or lane delimiters given the image.

3.1. Top-view projection

We briefly review Inverse Perspective Mapping (IPM). In short, IPM is a homography that gets a front view image and generates a virtual top view image as depicted in the images in Figure 1(a). It is equivalent to applying a camera rotation homography (view is rotated downwards) followed by an anisotropic scaling [10]. In our implementation we want to ensure that each pixel in the top view image corresponds to a predefined position on the road, *independent* of the camera intrinsics and its pose relative to the road.

See Figure 4 for an illustration of the following definitions. The camera coordinates $\mathcal{C}_{camera} = (\acute{x}, \acute{y}, \acute{z})$ are set such that $\acute{y}$ is the camera viewing direction. Let P_{road} be the plane *tangent* to the local road surface. We define the road coordinates $\mathcal{C}_{road} = (x, y, z)$ as follows: z direction is the normal to P_{road}, y is the projection of $\acute{y}$ onto P_{road} and the origin is the projection of the camera center onto P_{road}. Let T_{c2r} be the 6-D.O.F. transformation between $\mathcal{C}_{camera}$ and $\mathcal{C}_{road}$ (3D translation and 3D rotation). Since we assume

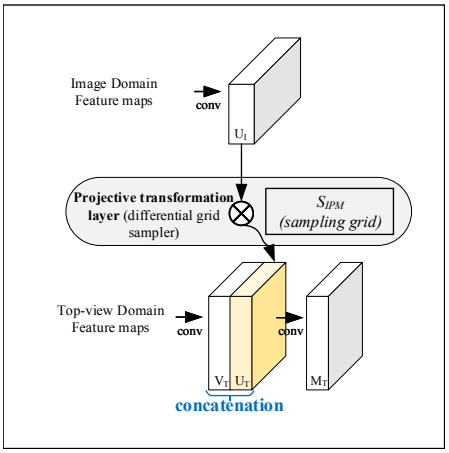

Figure 5. **The dual context module.** See section 3.2.

zero camera roll, T_{c2r} is uniquely defined by the camera tilt angle θ and its height above the ground h_{cam}. The homography $H_{r2i} : \mathbb{P}^2 \mapsto \mathbb{P}^2$, mapping each point on P_{road} to image plane coordinates, is determined by T_{c2r} and κ (See [10], Section 8.1.1). Finally, the IPM is obtained from H_{r2i} using a fixed set of parameters IPM_{Params} defining the top view region boundaries and an anisotropic scaling from meters to pixels. The top view image is generated using bilinear interpolation defined by a sampling grid S_{IPM}.

3.2. The projective transformation layer and the dual context module

A main building block in our architecture is the *projective transformation layer*. This layer is a specific realization, with slight variations, of the *spatial transformer module* [13]. It performs a differentiable sampling of an input feature map U_I, corresponding spatially to the *image plane*, to an output feature map U_T corresponding spatially to a *virtual top view* of the scene. The differential sampling is achieved through a grid generated as described in the previous section for transforming an image to top-view. The dual context module, illustrated in Figure 5, uses the projective transformation layer to create highly descriptive feature maps. Information flows from two multi-channel feature maps U_I and V_T corresponding to image-view and top-view respectively. Using the sampler described above, U_I is transformed to a corresponding multi-channel feature map U_T, and concatenated along the channel axis to V_T which shares the same spatial dimensions. This is the basic dual context module which can be repeated within the network. Consider now as an example a subsequent top view feature map M_T, obtained from the concatenated top view tensor $[V_T; U_T]$ by applying a convolution layer. M_T combines the following two desirable properties for lane detection. First, translational invariance in the top-view plane. This is valuable since in the top view lanes have similar appearance and geometry across the space. Second, preservation of a dual information context - in both image and top view. The additional image-view context encodes information which is not present in the top view such as fences, skyline and trees which are crucial for deciphering the 3D structure of the scene. Particularly, in the far range, the image-view context is much richer in visual information, and represents a much larger actual area compared to the top view.

3.3. Network structure

An overview of the 3D-LaneNet is illustrated in Figure 6. Information is processed in two parallel streams or pathways: the image-view pathway and the top-view pathway. We call this a *the dual-pathway backbone*. The image-view pathway processes and preserves information from the image while the top-view pathway provides the features with translation invariance and is used to predict the 3D lane detection output. The architecture for the image-view pathway is based on VGG16 [29] while the top-view pathway is similarly structured. Information flows to the top-view pathway through four projective transformation layers. To prevent exceeding the original number of channels in the top-view pathway feature maps, we add right after the projective transformation layer, another dimensionality reduction layer (1×1 convolution) in all the dual context modules except for the first one.

3.3.1 Road projection prediction branch

The first intermediate output of the image-view pathway network is an estimation of the "road projection plane" P_{road}. Essentially, this branch predicts T_{c2r}, the camera ($\mathcal{C}_{camera}$) to road ($\mathcal{C}_{road}$) transformation. It is trained in a supervised manner. T_{c2r} determines the top-view homography H_{r2i} and sampling grid S_{IPM} as explained in Section 3.1, and is therefore needed for the feed-forward step of the top-view pathway. At inference time, it is used also to translate the network output which is expressed in $\mathcal{C}_{road}$, back to $\mathcal{C}_{camera}$. As described in Section 3.1, T_{c2r} is defined in our case by the camera height h_{cam} and tilt θ, and therefore these are the two outputs of this branch.

3.3.2 Lane prediction head

At the heart of our end-to-end approach lies the **anchor-based lane representation**. Inspired by object detection, we use anchors to define lane candidates and a refined geometric representation to describe the precise 3D lane shape for each anchor. The output coordinate system is the estimation of $\mathcal{C}_{road}$ determined by h_{cam}, θ. Our anchors correspond to longitudinal lines in this coordinate system and the refined lane geometry to 3D points relative to the respective anchor. As illustrated in Figure 7, we define the anchors by equally spaced vertical (longitudinal) lines in x-positions

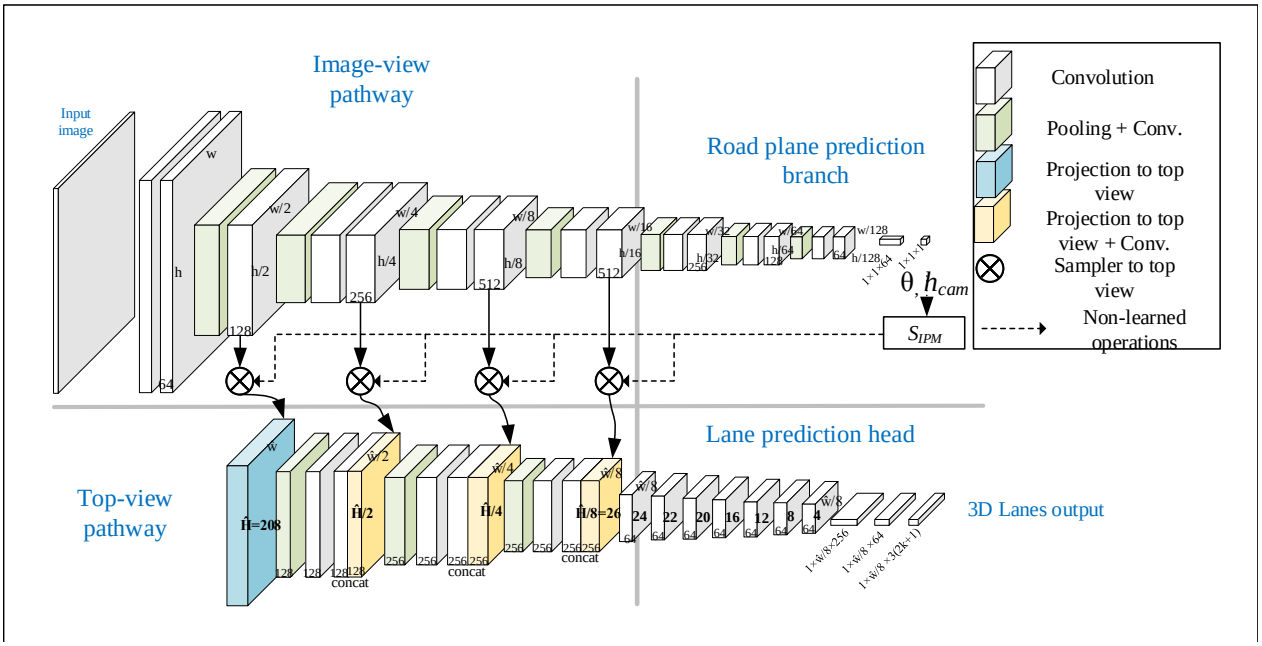

Figure 6. **3D-LaneNet network architecture.**

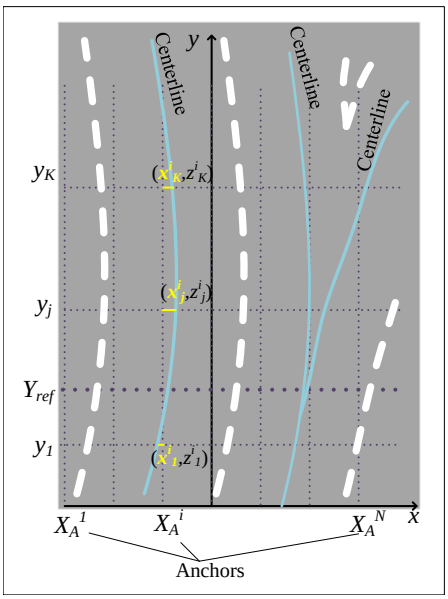

Figure 7. **Output representation.** Note that the number of anchors (N) equals the output layer width (marked by $\hat{w}/8$ in Fig. 6).

$\{X_A^i\}_{i=1}^N$. Per anchor X_A^i, a $3D$ lane is represented by $2 \cdot K$ output neurons activation $(\mathbf{x}^i, \mathbf{z}^i) = \{(x_j^i, z_j^i)\}_{j=1}^K$, which together with a fixed vector of K predefined y positions ($\mathbf{y} = \{y_j\}_{j=1}^K$) define a set of $3D$ lane points. The values x_j^i are horizontal offsets *relative to the anchor position* X_A^i. Meaning, the output (x_j^i, z_j^i) represents the point $(x_j^i + X_A^i, y_j, z_j^i) \in \mathbb{R}^3$, in C_{road} coordinates. In addition, for each anchor i, we output the confidence p^i that there is a lane associated with the anchor. We use a predefined longitudinal coordinate Y_{ref} for the association. The anchor X_A^i associated to a lane is the one closest to the x-coordinate of the lane at $y = Y_{ref}$.

Per anchor, the network outputs *three* types (t) of lane descriptors (confidence and geometry), the first two (c_1, c_2) represent lane centerlines and the third type (d) a lane delimiter. Assigning two possible centerlines per anchor yields the network support for merges and splits which may often result in having the centerlines of two lanes coincide at Y_{ref} and separating at different road positions as in the rightmost example in Figure 7. The topology of lane *delimiters* is generally more complicated compared to centerlines and our representation cannot capture all situations (for example the lane delimiters not crossing $y = Y_{ref}$ in Fig. 7). The prediction head of the 3D-LaneNet is designed to produce the described output. Through a series of convolutions with no padding in the y dimension, the feature maps are reduced in height, and finally the prediction layer size is $N \times 1 \times 3 \cdot (2 \cdot K + 1)$ s.t. each column $i \in \{1 \dots N\}$ corresponds to a single anchor X_A^i. Per anchor, X_A^i and type $t \in \{c_1, c_2, d\}$ the network output is denoted by $(\mathbf{x}_t^i, \mathbf{z}_t^i, p_t^i)$. The final prediction performs a 1D non-maximal suppression as common in object detection: only lanes which are locally maximal in confidence (compared to the left and right neighbor anchors) are kept. Each remaining lane, represented by a small number (K) of 3D points, is translated

to a smooth curve using spline interpolation.

3.4. Training and ground truth association

Given an image example and its corresponding 3D lane curves, $\{C_i\}_{i=1}^{N_C}$ (centerlines) and $\{D_i\}_{i=1}^{N_D}$ (delimiters), the training proceeds as follows. First, the ground truth coordinate system $\mathcal{C}_{road}$ is defined for the local road tangent plane as described in Sec. 3.1 using the known pitch ($\hat{\theta}$) and camera height ($\hat{h}_{cam}$). Next, each lane curve, projected to the $x-y$ plane of $\mathcal{C}_{road}$, is associated with the closest anchor at Y_{ref}. The leftmost lane delimiter and leftmost centerline associated with an anchor are assigned to the c_1 and d output types for that anchor. If an additional centerline is associated to the same anchor it is assigned to output type c_2. This assignment defines the ground truth (GT) per example in the same format as the output: per anchor X_A^i and type t the associated GT is denoted by $\left(\hat{\mathbf{x}}_t^i, \hat{\mathbf{z}}_t^i, \hat{p}_t^i\right)$, where $\hat{p}_t^i$ is an anchor/type assignment indicator, and the coordinates in $\mathcal{C}_{road}$.

Both in training time and in the evaluation, entire lanes are ignored if they do not cross Y_{ref} inside valid top-view image boundaries, and lane points are ignored if occluded by the terrain (i.e. beyond a hill top). The overall loss function of the network is given in Eq. 1. It combines three equally weighed loss terms: lane detection (Cross-entropy-loss), lane geometry and road plane estimation (L_1-loss).

$$\begin{aligned}\mathcal{L} = &- \sum_{t\in\{c_1,c_2,d\}} \sum_{i=1}^{N} \left(\hat{p}_t^i \log p_t^i + \left(1-\hat{p}_t^i\right) \log\left(1-p_t^i\right)\right) \\ &+ \sum_{t\in\{c_1,c_2,d\}} \sum_{i=1}^{N} \hat{p}_t^i \cdot \left(\left\|\mathbf{x}_t^i - \hat{\mathbf{x}}_t^i\right\|_1 + \left\|\mathbf{z}_t^i - \hat{\mathbf{z}}_t^i\right\|_1\right) \\ &+ \left|\theta - \hat{\theta}\right| + \left|h_{cam} - \hat{h}_{cam}\right| \end{aligned} \tag{1}$$

4. Experiments

Our experiments are presented as follows. We first review our method for generating a new synthetic dataset *synthetic-3D-lanes* which is used to derive most of our study conclusions. Next, we describe an evaluation method for the 3D lane detection task. We then present the results on the synthetic dataset and an ablation study examining the contribution of each concept in our overall approach. Finally, to prove the applicability of our approach to real data, we compare an image-only version of 3D-LaneNet to existing state-of-the-art methods on the tuSimple benchmark [1].

4.1. Synthetic 3D lane dataset

We generated the *synthetic-3D-lanes* dataset using the open source graphics engine *blender* [2]. Our programmatic approach allows us to randomize each of the modeled elements, from the 3D geometry of the scene to object types as illustrated in Figure 8. The process of generating *each scene* is composed of the following steps:

Terrain 3D. The terrain is modeled by a Mixture of Gaussians distribution with the number of Gaussians and their parameters randomized. Figure 8(a) shows an example of such a terrain.

Lane topology. Number of lanes on the main road is selected. Then we choose if there is a secondary road and the number of lanes in it. Depending on the later direction of the camera in the scene the junction of the secondary road is viewed as a merge or a split.

Lane curvature. The geometry of each road in top view is modeled by a third degree polynomial. In case a merge / split exists the junction point is chosen. This results in a top view lane-level map as shown in Figure 8(b). Lane width is randomly chosen between 3 and 4 meters.

Lane 3D. The top-view lane map is placed on the terrain, and secondary roads are lifted to simulate common road topography. 8(c) shows the result of this stage.

Terrain and road appearance. The texture of the road and the terrain are chosen from a set of textures. The type and color of the lane markings is also randomized.

Objects. Cars and trees are placed in the scene, on and off the road respectively. Their models are selected from a set of optional models.

Scene rendering. The host vehicle camera is positioned on the main road by choosing its lane and a lateral offset around lane center. The camera height is set randomly between 140cm and 190cm and a downward pitch between 0 and 5 degrees is selected. Finally, illumination is set and the scene is rendered from the camera view. The 3D points of each lane centerline and delimiter are translated to camera coordinates to generate the ground truth. Figure 8(d) includes several example scenes showing the resulting diversity and complexity.

Each generated example consists of an image (360×480 pixels) and its associated ground truth: 3D lanes, camera height and pitch. The generated dataset contains 300K train and 5K test examples. An additional 1K hold-out set was used for learning rate scheduling and choosing the best performing snapshot.

Implementation details. The image-view pathway of 3D-LaneNet is initialized from VGG16 trained on imagenet [6]. We train with Adam optimization [16] and with initial learning rate $5{\cdot}1e{-}4$. We use a variation on the cyclic learning rate regime described in [30] with a minimal learning rate of $1e{-}6$. The y-range of the top view representation is 80 meters and the x-range is 20 meters. The IPM scale is different in x and y: in the first top-view feature map each pixel corresponds to 16cm laterally (x) and 38.4cm longitudinally (y). The last top-view feature map is 8 times smaller

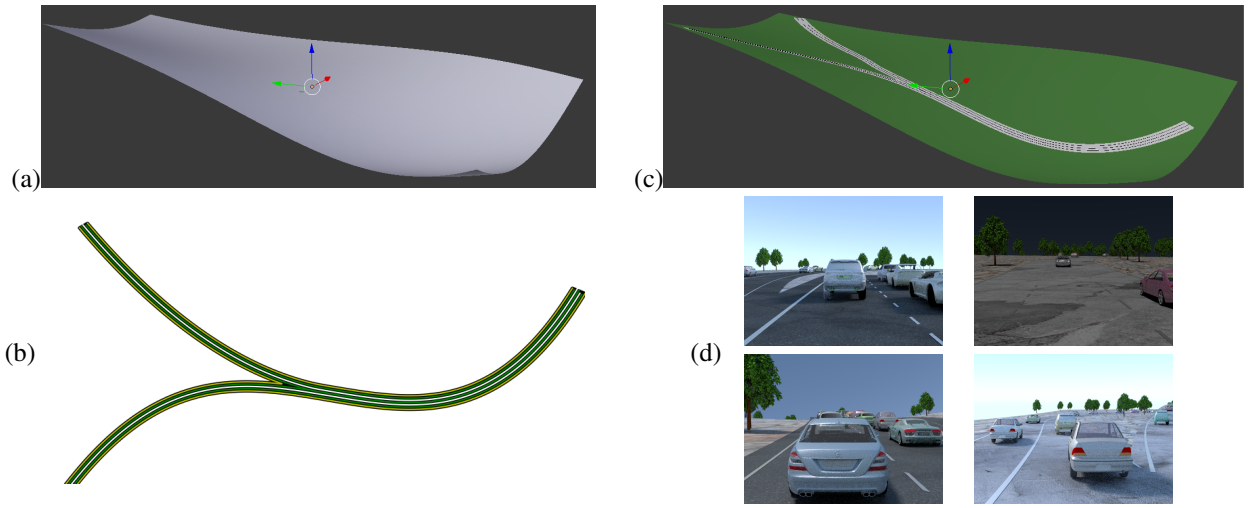

Figure 8. **Random synthetic data generation.** (a) Surface (b) Road topology and curvature (c) Road on surface (d) Rendered scenes.

and since there is an anchor per column the distance between anchors is $16 \times 8 = 128cm$. We set the $K(= 6)$ vertical reference points to be $\mathbf{y} = \{5, 20, 40, 60, 80, 100\}$ and $Y_{ref=20}$ meters.

4.1.1 Evaluation method

We propose an evaluation of 3D lane detection that separates the detection accuracy from the geometric estimation accuracy. *Detection accuracy* is computed via the standard average precision (AP) measure of the precision-recall curve. We first compute a curve to curve distance between a GT and a detected lane as a weighted sum of point-wise Euclidean distances. We measure distances on a set of predefined y-values along the curves, every 80cm in the range 0-80 meters. Weight is decreased for farther away points. We then perform a one-to-one (curve) matching by choosing pairs in decreasing similarity. A matching is considered correct if the weighted distance is below a certain, rather permissive, threshold (1.5 meters). Iterating over lane confidence thresholds we generate the precision-recall curve.

For matched detections we asses the *geometric estimation accuracy* by measuring the distribution of the error (point-wise Euclidean distance) over the same points used to measure the curve to curve distance. We further divide the entire dataset to lane points in the **near range** (0-30m) and **far range** (30-80m) due to the differences in the magnitude of errors. We then compute per range the 1σ error, as the 68 error percentile and the 2σ error as the 95 percentile. Lane centerline and delimiter detection are separately evaluated using this methodology. Irrelevant lane points are ignored in the evaluation as in the training phase.

4.1.2 Evaluation results

Typical network results on the test set are shown in Figure 2, with ground truth marked. The first row in Table 1 shows the quantitative results of 3D-LaneNet for centerline detection. A valid concern with synthetic datasets is that its variability is too limited and that the learned network memorizes the entire example space instead of learning to generalize. A positive indication that this is not the case is that test AP (0.952) is well below train AP (0.966) as are the geometric evaluation measures. In all the ablation tests presented next we initialized each network from the trained 3D-LaneNet weights to reduce training time.

We first examine the role of the ***dual-pathway architecture*** by comparing it to alternative architectures. The **image-view** only version connects the image-view pathway directly to the lane detection head which outputs the representation in $\mathcal{C}_{road}$ exactly as 3D-LaneNet does. The anchor positions X_A in this case are determined by the columns in the last feature map: For each column we pick a pixel at a predefined image y-coordinate and project it to top-view to determine the anchor corresponding to the column. The **top-view** only version first projects the *image* itself to top view and continues the same computation as the top-view pathway. In addition, we tested two versions which include a limited version of the dual-pathway. The **early IPM** includes a single dual context module (the first amongst the four in the full network). The **late IPM** similarly contains only the last dual context module out of the four. The results, summarized in table 1, show that the full dual-pathway architecture has superior performance compared to all other variants. In particular, the worst result is delivered **image-view** only version, stressing the importance of the top-view processing pathway. Note that the

late stage IPM, consisting of a trimmed version of the dual pathway, delivers the second best accuracy, but with a reduced computational cost, making it a good candidate for real-time implementations.

We tried several versions of the ***definition of the road projection plane*** which is in the heart of our architecture and output representation. Our first attempt was to learn the best road projection plane per scene without explicit supervision, similar to the "H-net" horizon estimation in [23], but this approach failed to produce satisfying results. We believe this direction deserves further study as it is the most natural one. Another natural choice is to take into consideration the entire scene when fitting the road plane and not only the local road normal. For this we devised a ground truth generation algorithm which takes the farthest visible road point and connects it to the local road position to determine the pitch. This method, is termed **horizon** in Table 1 since it resembles horizon estimation methods. Evidently, it performed slightly worse in general, although we observed consistently cases in which the scene topography favors this definition. We also tried assuming a **fixed position** of the camera, in which the average pitch (2.5°) and camera height (165cm) were used to define T_{c2r}.

The last row in Table 1 shows the result of the existing approach to ***image-to-world*** translation which assumes a **flat ground**. It is emulated by setting the elevation component (z) to zero in our best network result. Although the poor result is somewhat trivial and highly data dependent, it shows the importance of full 3D estimation. With 0.966 AP, the performance of 3D-LaneNet for lane *delimiter* detection surpasses that for the centerline, probably due to the explicit realization of delimiters in the image. This is apparent also in the positional error distribution (near range: 10.5cm@1σ, 28cm@2σ; far range: 27.3cm@1σ and 106cm@2σ).

Since output is transformed from road to camera coordinates using an estimated T_{c2r} we also measured the quality of this estimation and its effect on the results. The median values of the absolute errors for pitch (θ) and camera height (h_{cam}) are 0.09° and $2.4cm$ respectively. To eliminate the contribution of this error we evaluated performance in road coordinates C_{road} by taking the raw network output (before transforming to C_{camera}) and got a negligible difference in measured performance.

4.2. Results on the tuSimple benchmark

The tuSimple lane dataset [1] consists of 3626 training and 2782 test images and is currently the only large scale lane detection benchmark. In addition, today there is no access to the labels for the test images. We therefore divided the original training set to our own train/validation dataset (90% train and 10% validation). To cope with data which lacks 3D information we use a variation of the 3D-LaneNet. Instead of a 3D representation, the network output was reduced to 2D points on the road projection plane by eliminating the elevation ($\mathbf{z}_t^i$) component. Only the delimiter output type is maintained ($t = d$) since the marked entities in the dataset are lane delimiters. A fixed homography, $H_{tuSimple}$,between image plane and road projection plane, was manually selected, such that straight lanes become parallel in top view. The lanes directly predicted by the network are transformed to lanes in the image view using $H_{tuSimple}$. The road projection plane prediction branch is not used here. Other than the aforementioned, the network is identical to the 3D-LaneNet as configured for the synthetic-3D-lanes dataset. The tuSimple main evaluation metric (*acc*) is the average ratio of detected ground truth points per image. Using our end-to-end approach on our validation set we reached an accuracy of 0.951 which is a result competitive with the one achieved by the tuSimple 2017 competition winning method [25], (0.965). This result is encouraging and somewhat surprising given that our entire approach was designed towards the 3D estimation task. In particular, our geometric loss (Eq. 1) is computed in top view coordinates, giving in practice a much higher weight to distant lane points while in the tuSimple acc metric all points equally contribute.

Table 1. Centerline detection results on *synthetic-3D-lanes* dataset

	AP	Error **near** (cm)		Error **far** (cm)	
		1σ	2σ	1σ	2σ
3D-LaneNet	**0.952**	**10.8**	**29.1**	**30**	**118**
image-view	0.904	22.5	56	86.5	276
top-view	0.917	20.1	42.6	64	263
early IPM	0.926	14.9	36.6	50.3	231
late IPM	0.943	14.7	37.8	40.6	158
horizon	0.951	12	34	32.2	130
fixed position	0.944	12.3	34.2	35.6	144
flat ground	0.765	20.1	74.5	148	453

5. Conclusions and future work

We presented a network capable of single-frame lane detection with one CNN feed-forward operation requiring no additional post processing. We showed competitive results on the tuSimple benchmark and its capability to estimate full 3D coordinates of multiple lanes on a newly introduced synthetic dataset. The most important following study is to prove its ability to detect 3D lanes on real data as well. Our current representation assumes roughly longitudinal lanes which limits the applicability of the method in complex topologies like urban intersections. In future work we would like to adapt our representation to handle such cases as well, using an end-to-end training scheme.

References

[1] http://benchmark.tusimple.ai/, lane challenge.

[2] https://www.blender.org/.

[3] M. Aly. Real time detection of lane markers in urban streets. In *IVS*, pages 7–12, 2008.

[4] A. Bar Hillel, R. Lerner, D. Levi, and G. Raz. Recent progress in road and lane detection: a survey. *Machine Vision and Applications*, 25(3):727–745, Apr 2014.

[5] A. Borkar, M. Hayes, and M. T. Smith. Robust lane detection and tracking with ransac and kalman filter. In *ICIP*, pages 3261–3264, Nov 2009.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

[7] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[8] M. Ghafoorian, C. Nugteren, N. Baka, O. Booij, and M. Hofmann. El-gan: Embedding loss driven generative adversarial networks for lane detection. *CoRR*, abs/1806.05525, 2018.

[9] R. Gopalan, T. Hong, M. Shneier, and R. Chellappa. A learning approach towards detection and tracking of lane markings. *IEEE Transactions on Intelligent Transportation Systems*, 13(3):1088–1098, Sept 2012.

[10] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[11] B. He, R. Ai, Y. Yan, and X. Lang. Accurate and robust lane detection based on dual-view convolutional neutral network. In *IVS*, pages 1041–1046, June 2016.

[12] B. Huval, T. Wang, S. Tandon, J. Kiske, W. Song, J. Pazhayampallil, M. Andriluka, P. Rajpurkar, T. Migimatsu, R. Cheng-Yue, F. Mujica, A. Coates, and A. Y. Ng. An empirical evaluation of deep learning on highway driving. *CoRR*, abs/1504.01716, 2015.

[13] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. In *NIPS*, pages 2017–2025, 2015.

[14] J. Kim and M. Lee. Robust lane detection based on convolutional neural network and random sample consensus. In *Neural Information Processing*, pages 454–461. Springer, 2014.

[15] J. Kim and C. Park. End-to-end ego lane estimation based on sequential transfer learning for self-driving cars. In *CVPR Workshops*, pages 1194–1202, July 2017.

[16] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[17] S. Lee, J. Kim, J. Shin Yoon, S. Shin, O. Bailo, N. Kim, T.-H. Lee, H. Seok Hong, S.-H. Han, and I. So Kweon. Vpgnet: Vanishing point guided network for lane and road marking detection and recognition. In *CVPR*, pages 1947–1955, 2017.

[18] J. Li, X. Mei, D. Prokhorov, and D. Tao. Deep neural network for structural prediction and lane detection in traffic scene. *IEEE Transactions on Neural Networks and Learning Systems*, 28(3):690–703, March 2017.

[19] F. Liu, C. Shen, and G. Lin. Deep convolutional neural fields for depth estimation from a single image. In *CVPR*, pages 5162–5170, 2015.

[20] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, and A. Berg. Ssd: Single shot multibox detector. In B. Leibe, J. Matas, M. Welling, and N. Sebe, editors, *ECCV*, pages 21–37, Germany, 1 2016. Springer.

[21] H. A. Mallot, H. H. Bülthoff, J. J. Little, and S. Bohrer. Inverse perspective mapping simplifies optical flow computation and obstacle detection. *Biological Cybernetics*, 64(3):177–185, Jan 1991.

[22] A. Meyer, N. Salscheider, P. Orzechowski, and C. Stiller. Deep semantic lane segmentation for mapless driving. In *IROS*, Oct 2018.

[23] D. Neven, B. D. Brabandere, S. Georgoulis, M. Proesmans, and L. V. Gool. Towards end-to-end lane detection: an instance segmentation approach. *CoRR*, abs/1802.05591, 2018.

[24] G. L. Oliveira, W. Burgard, and T. Brox. Efficient deep models for monocular road segmentation. In *IROS*, pages 4885–4891, Oct 2016.

[25] X. Pan, J. Shi, P. Luo, X. Wang, and X. Tang. Spatial as deep: Spatial CNN for traffic scene understanding. In *AAAI*, pages 7276–7283, 2018.

[26] D. Pomerleau. Ralph: rapidly adapting lateral position handler. In *IVS*, pages 506–511, Sept 1995.

[27] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, pages 779–788, 2016.

[28] S. Shah, D. Dey, C. Lovett, and A. Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, 2017.

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

[30] L. N. Smith. Cyclical learning rates for training neural networks. In *WACV*, pages 464–472, March 2017.

[31] C. Urmson, J. Anhalt, D. Bagnell, C. R. Baker, R. Bittner, M. N. Clark, J. M. Dolan, D. Duggins, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. M. Howard, S. Kolski, A. Kelly, M. Likhachev, M. McNaughton, N. Miller, K. M. Peterson, B. Pilnick, R. Rajkumar, P. E. Rybski, B. Salesky, Y. Seo, S. Singh, J. M. Snider, A. Stentz, W. Whittaker, Z. Wolkowicki, J. Ziglar, H. Bae, T. Brown, D. Demitrish, B. Litkouhi, V. Sadekar, W. Zhang, J. Struble, M. Taylor, M. Darms, and D. Ferguson. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.