Eigenlanes: Data-Driven Lane Descriptors for Structurally Diverse Lanes

Dongkwon Jin\textsuperscript{1}, Wonhui Park\textsuperscript{1}, Seong-Gyun Jeong\textsuperscript{2}, Heeyeon Kwon\textsuperscript{2}, Chang-Su Kim\textsuperscript{1}
Korea University\textsuperscript{1}, 42dot.ai\textsuperscript{2}
{dongkwonjin, whp}@mcl.korea.ac.kr, {seonggyun.jeong, heeyeon.kwon}@42dot.ai, changsukim@korea.ac.kr

Abstract

A novel algorithm to detect road lanes in the eigenlane space is proposed in this paper. First, we introduce the notion of eigenlanes, which are data-driven descriptors for structurally diverse lanes, including curved, as well as straight, lanes. To obtain eigenlanes, we perform the best rank-$M$ approximation of a lane matrix containing all lanes in a training set. Second, we generate a set of lane candidates by clustering the training lanes in the eigenlane space. Third, using the lane candidates, we determine an optimal set of lanes by developing an anchor-based detection network, called SIIC-Net. Experimental results demonstrate that the proposed algorithm provides excellent detection performance for structurally diverse lanes. Our codes are available at https://github.com/dongkwonjin/Eigenlanes.

1. Introduction

Lane detection is essential for understanding driving environments, in which autonomous or human drivers should abide by traffic rules. To control vehicle maneuvers, boundaries of road lanes and sidewalks should be detected reliably. There are various challenging factors to interfere with the detection of those lanes. For example, lanes may be unobvious or even invisible due to weather and illumination conditions or due to the occlusion by nearby vehicles, as illustrated in Figure 1(a).

For lane detection, traditional methods extract handcrafted features, such as image gradients or color features [2, 10, 11, 35]. This feature engineering may fail to discriminate actual lanes from noisy ones. Recently, various CNN-based techniques have been developed to detect lanes in real environments more reliably. Most such techniques adopt the semantic segmentation framework [5, 12, 13, 21, 22, 34], in which each pixel in an image is dichotomized into either lane or no-lane category. To preserve continuous lane structure in detection results, several attempts have been made, including curve fitting [21], polynomial regression [32], and adversarial training [5]. However, even these algorithms may fail to detect less visible lanes in cluttered scenes, because they use only local features and may miss parts of lanes. Meanwhile, the anchor-based detection framework, which has been used extensively in object detection [15, 20, 27, 28], has been recently adopted for lane detection [18, 30]. These anchor-based algorithms consider straight lines as anchors (or lane candidates). Then, they declare each anchor as a lane or not. By exploiting long-range contextual information, they can detect implied lanes effectively. However, because of the straight lane assumption, it may not deal with complicated lanes, such as curved and winding ones in Figure 1(b).

In this paper, we propose a novel algorithm to detect structurally diverse road lanes in the eigenlane space. It enables to process curved, as well as straight, lanes reliably in the anchor-based detection framework. First, we introduce the notion of eigenlanes, which are data-driven lane descriptors. To obtain eigenlanes, we approximate a lane matrix, which contains all lanes in a training set, based on the low-rank approximation property of singular value decomposition (SVD) [4]. Then, each lane is represented by a linear combination of $M$ eigenlanes. Second, we generate a set of lane candidates, including complicated and curved ones, by clustering the training lanes in the eigenlane space. Third, we develop an anchor-based detector, called SIIC-Net, to detect lanes from the candidates. It consists of two modules:
self-lane identification (SI) module and inter-lane correlation (IC) module. SI computes the classification probability and regression offset of each candidate, while IC estimates the compatibility between each pair of lanes. Extensive experiments show that the proposed algorithm provides competitive results on existing datasets [1, 22] and outperforms the state-of-the-art techniques [30, 34] on a new dataset, called SDLane, containing structurally more diverse lanes.

This work has the following major contributions:

- We propose the notion of eigenlanes, which are data-driven lane descriptors, to represent structurally diverse lanes compactly in the eigenlane space.
- We develop SIIC-Net to detect and regress road lanes in the eigenlane space effectively and efficiently. It yields outstanding performances on various datasets.
- We construct the SDLane dataset to represent structurally diverse and complicated lanes in real driving environments more faithfully than the existing datasets do.1

2. Related Work

In autonomous driving systems, it is required to detect boundaries of road lanes, sidewalks, or crosswalks precisely and reliably. Whereas early methods [2, 10, 11, 35] adopted hand-crafted low-level features, several CNN-based lane detectors have been developed recently to cope with complicated road scenes using deep features. Most of these techniques are based on the semantic segmentation framework [5, 12, 13, 21, 22, 34], in which pixel-wise classification is performed to decide whether each pixel belongs to a lane or not. In [22], Pan et al. developed a convolutional network to propagate spatial information between pixels through message passing. Zheng et al. [34] passed the information more efficiently using a recurrent feature aggregation module. In [13], Hou et al. proposed a self-attention distillation mechanism to train the network more effectively. Also, Hou et al. [12] employed teacher and student networks to transfer structural relationships between lanes by constructing an inter-region affinity graph. To maintain long-range consistency of segmented results, Ghafoorian et al. [5] used a discriminator to refine prediction results of a generator through adversarial training. In [21], Neven et al. applied a perspective transformation to segmented pixels of each lane and used the transformed points for polynomial fitting.

Alternative approaches, different from the segmentation framework, also have been developed. In [24], a network predicts the probability that vertically neighboring pixels belong to the same lane. Then, through greedy iterations, trajectories of pixels are concatenated to form a full lane.

1SDLane is available at https://www.42dot.ai/akit/dataset.

3. Proposed Algorithm

We propose a novel algorithm to detect structurally diverse road lanes in the eigenlane space. Figure 2 shows an overview of the proposed algorithm. First, the eigenlane space is constructed by performing the low-rank approximation of lanes in a training set. Second, lane candidates are generated by clustering lanes in the eigenlane space. Third, given an image, an optimal set of lanes are determined from the lane candidates by SIIC-Net.

3.1. Eigenlanes – Formulation

SVD and principal component analysis (PCA) are used in various fields to represent data compactly [4, 16]. A well-known such application is face recognition using eigenfaces [31]. Also, in this conference, eigencontours [23] are proposed to describe object boundaries. In this paper, we use SVD to represent road lanes. Specifically, we adopt a data-driven approach and exploit the distribution of lanes in a training set, instead of employing parametric curves such as polynomials [29] or splines [3, 6], to represent lanes.

Definition of eigenlane space: A lane can be represented by 2D points sampled uniformly in the vertical direction. Specifically, let $\mathbf{x} = [x_1, x_2, \ldots, x_N]^T$ be a lane, where $x_i$
is the $x$-coordinate of the $i$th sample and $N$ is the number of samples. We construct a lane matrix $A = [x_1, x_2, \cdots, x_L]$ from a training set containing $L$ lanes. Then, we apply SVD to the lane matrix $A$ by

$$A = U\Sigma V^\top$$

where $U = [u_1, \cdots, u_N]$ and $V = [v_1, \cdots, v_L]$ are orthogonal matrices and $\Sigma$ is a diagonal matrix, composed of singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$. Here, $r$ is the rank of $A$. It is known that

$$A_M = [\tilde{x}_1, \cdots, \tilde{x}_L] = \sigma_1 u_1 v_1^\top + \cdots + \sigma_M u_M v_M^\top$$

is the best rank-$M$ approximation of $A$ in that the Frobenius norm $\|A - A_M\|_F$ is minimized [4]. Also, the sum of squared lane approximation errors is given by

$$\|A - A_M\|^2_F = \sum_{i=1}^{L} \|x_i - \tilde{x}_i\|^2 = \sum_{i=M+1}^{r} \sigma_i^2.$$  

(3)

In (2), each approximate lane $\tilde{x}_i$ is given by a linear combination of the first $M$ left singular vectors $u_1, \cdots, u_M$. In other words,

$$\tilde{x}_i = U_M c_i = [u_1, \cdots, u_M] c_i.$$  

(4)

We refer to these $u_1, \cdots, u_M$ as eigenlanes, because they are eigenvectors of $AA^\top$. They can be regarded as principal components in PCA. However, strictly speaking, this is not PCA, since the mean lane is not removed in constructing $A$ [16]. We do not remove the mean (or center the data) because we are interested in the best low-rank approximation, instead of finding the best fitting subspace [4].

We call the space spanned by $\{u_1, \cdots, u_M\}$ as the eigenlane space. Given a lane $x$, we project it onto the eigenlane space to obtain the approximation

$$\tilde{x} = U_M c.$$  

(5)

where the coefficient vector $c$ is given by

$$c = U_M^\top x.$$  

(6)

**Algorithm 1** Lane candidate generation in eigenlane space

**Input:** Set of training lanes $\{x_1, x_2, \cdots, x_L\}$, $M = \#$ of eigenlanes, $K = \#$ of lane candidates

1. Construct the lane matrix $A$ and perform SVD in (1)
2. Transform each lane $x_i$ to $c_i$ via (6)
3. Apply the $K$-means algorithm [8] to $\{c_1, c_2, \cdots, c_L\}$ to obtain $K$ centroids $c^1, \cdots, c^K$.
4. Generate the lane candidate $I_k = U_M c^k$ by inversely transforming each centroid $c^k$ via (5)

**Output:** Set of lane candidates $\{I_1, \cdots, I_K\}$

Thus, in the eigenlane space, a lane $x$ is approximately represented by the $M$-dimensional vector $c$ in (6). Also, the approximate $x$ can be reconstructed from $c$ via (5).

**Detection and regression in eigenlane space:** Let $\{\tilde{x}_1, \cdots, \tilde{x}_L\}$ be the set of training lanes, which are already approximated via (4). By clustering these lanes, we obtain a finite number of lane candidates (or anchors) for lane detection. However, instead of the original lane space of dimension $N$, we perform the clustering in the eigenlane space of dimension $M$, as illustrated in Figure 3. This is possible because the transform $U_M$ is length-preserving:

$$\|\tilde{x}_i - \tilde{x}_j\| = \|c_i - c_j\|.$$  

(7)

Also, $M < N$. The clustering in the lower-dimensional space is more effective and more efficient. **Algorithm 1** summarizes the process of lane candidate generation.

Suppose that a lane candidate $I = U_M c$ is detected. Then, we refine it to

$$I + \Delta I = U_M (c + \Delta c),$$  

(8)

by finding an offset vector $\Delta c$ using a regressor. This is also done in the eigenlane space, since $\|\Delta I\| = \|\Delta c\|$.

### 3.2. Eigenlanes – Image examples

**Eigenlanes in image space:** In this example, we use the TuSimple dataset [1] to determine eigenlanes. Here, each lane is represented by a 50D vector, i.e. $N = 50$. Figure 4(a) shows the four eigenlanes $u_1, u_2, u_3, u_4$, which are sufficient to represent all TuSimple lanes faithfully. The first eigenlane $u_1$ is a slanted line, instead of a vertical line. Most lanes are slanted from the viewpoint of a driving car. Because $u_1$ achieves the best rank-1 approximation of these lanes, it is also slanted. By weighting $u_1$, we can represent straight road lanes in an image. Next, $u_2$ is slightly curved at the top side (far from the cameras), and $u_3$ is curvier. These eigenlanes are required to represent simply curved lanes. Finally, $u_4$ has an inflection point and is used to describe highly complicated lanes.

In Figure 4(b), the straight line parts of the left and middle lanes are slanted to the right, whereas $u_1$ is slanted to
the left. Thus, the coefficients for \( u_1 \) for these two lanes are negative. For all lanes, the coefficients for \( u_2 \) and \( u_3 \) are not negligible because the lanes are curved. They are, however, not complicated, so their 4th coefficients are insignificant.

**Rank-M approximation:** Figure 5 shows examples of original lanes and their rank-\( M \) approximations. In (b), using only one eigenlane \( u_1 \), the rank-1 approximation yields line parts of the lanes. In (c), the rank-2 approximation reconstructs curved parts additionally. To represent the curved parts more faithfully, the rank-3 approximation in (d) is required, which matches well the ground-truth in (a).

**Lane candidates:** As mentioned previously, we generate lane candidates for detection, by grouping training lanes in the eigenlane space using the K-means algorithm. Figure 6 shows such generated candidates according to \( K \). Being the centroids, they are representative of all training lanes. Notice that the proposed SDLane dataset contains many curved lanes with high curvatures, whereas the existing CULane dataset [22] consists of mainly straight lanes. TuSimple contains curved lanes, which, however, lack diversity.

### 3.3. SIIC-Net

Using the \( K \) lane candidates \( \{1, \ldots, K\} \), the proposed SIIC-Net detects road lanes. In Figure 7, SIIC-Net consists of the encoder-decoder part, SI module, and IC module. After the SI module, the non-maximum suppression (NMS) is performed to filter out redundant candidates. Also, after the IC module, the maximum weight clique selection (MWCS) is done to determine an optimal set of lanes.

**Encoder-decoder part:** We adopt ResNet50 [9] as the encoder to extract features and employ the auxiliary branch [25] as the decoder to yield a binary segmentation map of lanes. From an image, we extract multi-scale feature maps and aggregate the three lowest-level maps. To this end, we match the resolutions of the two smaller maps to the finest one via bilinear interpolation. Let \( X_s = [X^1_s, X^2_s, \ldots, X^K_s] \in \mathbb{R}^{H \times W \times C_1} \) be the aggregated feature map, where \( H, W \), and \( C_1 \) are the feature height, the feature width, and the number of channels. Then, we squeeze \( X_s \) using convolutional layers to yield \( X_a \in \mathbb{R}^{H \times W \times C_2} \). The decoder processes \( X_a \) to produce the segmentation map. We use the decoder part in the training phase only, as in [25].

**Self-lane identification (SI) module:** For each lane candidate \( l_k \), we estimate the lane probability, the positional offset, and the height of the topmost point using the SI module, the structure of which is in Figure 8(a). The SI module employs a line pooling layer [7, 17]. From the squeezed feature map \( X_s \), it obtains the lane feature map \( Y_c = [Y^1_c, Y^2_c, \ldots, Y^K_c] \in \mathbb{R}^{C_2 \times C_2} \) by averaging the features of pixels along \( l_k \):

\[
Y^c_k(k) = \frac{1}{|l_k|} \sum_{p \in l_k} X^c_s(p) \quad (9)
\]

for \( 1 \leq k \leq K \) and \( 1 \leq c \leq C_2 \), where \(|l_k|\) denotes the number of pixels in \( l_k \). Then, two probability vectors and a lane offset matrix are obtained by

\[
P = \sigma(f_1(Y)), \quad H = \sigma(f_2(Y)), \quad O = f_3(Y) \quad (10)
\]

where \( f_1 \) and \( f_2 \) are fully-connected layers of sizes \( C_2 \times 2 \) and \( C_2 \times R \) for classification, \( f_3 \) is a fully-connected layer of size \( C_2 \times M \) for regression, and \( \sigma(\cdot) \) is the softmax function. For lane candidate \( l_k \), \( P_k \) informs the probabilities that it is a lane or not, \( H_k \) represents the probabilities that its ending point is located at one of \( R \) pre-defined heights, and \( O_k = \Delta c_k \) is an offset vector in (8) for lane refinement.

**NMS:** Many redundant lanes tend to be detected around an actual one. We filter out those overlapping ones through
Figure 7. The architecture of the proposed SIIC-Net: Given an image, the encoder extracts two types of feature maps and the decoder yields a segmentation map. Then, the self-lane identification module (SI) and the inter-lane correlation (IC) module process the feature maps. After the SI module, NMS removes redundant lane candidates. After the IC module, MWCS determines an optimal set of lanes.

Figure 8. Block diagrams of the SI and IC modules.

Figure 9. An example of the lane detection by SIIC-Net: (a) input image, (b) 10 selected lanes after NMS, (c) optimal lanes determined by MWCS, and (d) refined lanes using the regression offsets from the SI module.

**MWCIS:** We determine an optimal set of lanes by employing MWCS [14], which is a graph optimization technique. We first construct a complete graph $G = (V, E)$. The node set $V = \{v_1, v_2, \ldots, v_T\}$ represents the $T$ selected lanes from NMS. Every pair of lanes is connected by an edge in the edge set $E = \{(v_i, v_j) : i \neq j\}$. Each edge is assigned weight $w(v_i, v_j) = \frac{R_{i,j} + R_{j,i}}{2}$.

Let $\theta$ denote a clique, represented by the index set of member nodes. We define the compatibility $E_{\text{compatible}}(\theta)$ of clique $\theta$ as

$$E_{\text{compatible}}(\theta) = \sum_{i \in \theta} \sum_{j \in \theta, j \neq i} w(v_i, v_j).$$

(13)

We then select the maximal weight clique $\theta^*$ by

$$\theta^* = \arg \max_{\theta} E_{\text{compatible}}(\theta)$$

(14)

subject to a constraint $w(v_i, v_j) > \kappa$ for all edges in the clique, where $\kappa$ is a threshold. If there is no clique satisfying the constraint, we select the maximal single-node clique $\theta^* = \{i^*\}$, where $i^* = \arg \max_i P_{v_i}$.

Next, we refine each lane in the optimal clique $\theta^*$ by $U(c_{v_i} + \Delta c_{v_i})$, where $\Delta c_{v_i}$ is the offset vector in the eigen-lane space, predicted by the SI module. Moreover, we refine the vertical height of each lane, by removing sampled points whose $y$-coordinates are bigger than $H_{v_i}$. Figure 9(c) and (d) show the MWCS results and their refined ones.

The supplemental document (Section A) describes the training process and the architecture of SIIC-Net in detail.
4. Experimental Results

4.1. Datasets

TuSimple [1]: It consists of 6,408 images only, which are split into 3,268 training, 358 validation, and 2,782 test images. For each image, lanes are annotated by the 2D coordinates of sampling points with a uniform height interval of 10 pixels. It contains both straight and curved lanes, whose shapes are, however, simple and similar to one another.

CULane [22]: It is a rich dataset with about 130K images. Its 34,680 test images are classified into 9 categories. In some categories, lanes are highly implied or even invisible. For each image, pixel-wise masks for up to 4 road lanes are provided. Most lanes are straight lines.

SDLane: We construct a structurally diverse lane dataset SDLane. It contains highly curved and complicated lanes, as illustrated in Figure 10. We collect 43K images, which are split into about 39K training and 4K testing images, and annotate the actual lanes manually. As mentioned in Section 3.2 and Figure 6, SDLane contains more curved lanes than CULane and more diverse lanes than TuSimple. The structural diversity of SDLane is discussed in detail in the supplemental document (Section B).

4.2. Evaluation metrics

In TuSimple, a lane point is regarded as correctly located if its distance to the ground-truth point is shorter than a threshold [1]. Then, the accuracy is defined as \( \frac{N_c}{N} \), where \( N \) is the number of ground-truth lane points, and \( N_c \) is the number of correctly predicted lane points. Also, the false positive rate (FPR) and the false negative rate (FNR) are

\[
\text{FPR} = \frac{F_{\text{pred}}}{N_{\text{gt}}} \quad \text{FNR} = \frac{M_{\text{pred}}}{N_{\text{gt}}} \tag{15}
\]

where \( F_{\text{pred}} \) is the number of incorrectly predicted lanes, \( N_{\text{pred}} \) is that of predicted lanes, \( M_{\text{pred}} \) is that of missed lanes, and \( N_{\text{gt}} \) is that of ground-truth lanes.

In CULane and SDLane, each lane is regarded as a thin stripe with 30 pixel width [22]. A predicted lane is declared to be correct if its IoU ratio with the ground-truth is greater than 0.5. The precision and the recall are computed by

\[
\text{Precision} = \frac{TP}{TP+FP}, \quad \text{Recall} = \frac{TP}{TP+FN} \tag{16}
\]

where TP is the number of correctly detected lanes, FP is that of false positives, and FN is that of false negatives. Then, the F-measure is computed by

\[
\text{F-measure} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{17}
\]

4.3. Comparative assessment

Comparison on TuSimple: Table 1 compares the proposed algorithm with the conventional road lane detectors [13, 18, 21, 22, 25] on TuSimple. The proposed algorithm yields a high FNR, resulting in a relatively low accuracy, but provides the best FPR performance. Figure 11 shows some detection results. Most errors are caused by the lanes, which are far from the camera and thus short. Except for them, the proposed algorithm detects most lanes precisely, especially ego and alternative lanes, which are more important for driving.

Comparison on CULane: Table 2 compares the F-measure performances on CULane, whose lanes are divided into 9 categories. The proposed algorithm outperforms all conventional algorithms. Especially, the proposed algorithm yields excellent results on the challenging categories of ‘Night,’ ‘No line,’ and ‘Dazzle’ in which lanes are highly implicit or even invisible. This indicates that the proposed algorithm can detect challenging lanes by considering the correlation or compatibility among detected lanes. The performance gap against LaneATT [30] is marginal. But, when the same backbone of ResNet18 is used for both algorithms, the gap increases further. On the ‘Curve’ category, the proposed algorithm is inferior to the other methods. However, it does not mean that the proposed algorithm is not capable of detecting curved lanes. Since the proportion of curved lanes is only 1.2%, the CULane training data are not enough to generate curved lane candidates in the eigenlane space.

Table 1. Comparison on TuSimple. Only the algorithms with publicly available source codes are compared.

<table>
<thead>
<tr>
<th></th>
<th>Accuracy</th>
<th>FPR</th>
<th>FNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>LaneNet</td>
<td>96.38</td>
<td>0.0780</td>
<td>0.0244</td>
</tr>
<tr>
<td>SCNN</td>
<td>96.53</td>
<td>0.0617</td>
<td>0.0180</td>
</tr>
<tr>
<td>SAD</td>
<td>96.64</td>
<td>0.0602</td>
<td>0.0205</td>
</tr>
<tr>
<td>UFast</td>
<td>95.82</td>
<td>0.1903</td>
<td>0.0392</td>
</tr>
<tr>
<td>RESA</td>
<td>96.82</td>
<td>0.0363</td>
<td>0.0248</td>
</tr>
<tr>
<td>LaneATT</td>
<td>95.63</td>
<td>0.0353</td>
<td>0.0292</td>
</tr>
<tr>
<td>Proposed</td>
<td>95.62</td>
<td>0.0320</td>
<td>0.0399</td>
</tr>
</tbody>
</table>

Figure 10. Example images and ground-truth lanes in the SDLane dataset. Since crossroad images are included in SDLane, some lanes for left or right turns are highly curved and implicit.

Figure 11. Detection results of the proposed algorithm on the TuSimple dataset. Detected lanes are depicted in green, while false negatives are in red.
Table 2. Comparison of the F-measure performances (%) on the CULane dataset, whose lanes are classified into 9 categories. For the ‘Crossroad’ category, only FP is reported. * means that the encoder backbone is ResNet18.

<table>
<thead>
<tr>
<th>Category</th>
<th>Normal</th>
<th>Crowded</th>
<th>Night</th>
<th>No line</th>
<th>Shadow</th>
<th>Arrow</th>
<th>Dazzle</th>
<th>Curve</th>
<th>Crossroad</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCNN [22]</td>
<td>90.6</td>
<td>69.7</td>
<td>66.1</td>
<td>43.4</td>
<td>66.9</td>
<td>84.1</td>
<td>58.5</td>
<td>65.7</td>
<td>1990</td>
<td>71.6</td>
</tr>
<tr>
<td>SAD [13]</td>
<td>90.7</td>
<td>70.0</td>
<td>66.3</td>
<td>43.5</td>
<td>67.0</td>
<td>84.4</td>
<td>59.9</td>
<td>65.7</td>
<td>2052</td>
<td>71.8</td>
</tr>
<tr>
<td>UFast [25]</td>
<td>90.7</td>
<td>70.2</td>
<td>66.7</td>
<td>44.4</td>
<td>69.3</td>
<td>85.7</td>
<td>59.5</td>
<td>69.7</td>
<td>2037</td>
<td>72.3</td>
</tr>
<tr>
<td>Curve-Nas [33]</td>
<td>92.1</td>
<td>73.1</td>
<td>69.9</td>
<td>47.7</td>
<td>70.1</td>
<td>85.8</td>
<td>67.7</td>
<td>68.4</td>
<td>1746</td>
<td>74.8</td>
</tr>
<tr>
<td>LaneATT* [30]</td>
<td>91.1</td>
<td>73.0</td>
<td>69.0</td>
<td>48.4</td>
<td>70.9</td>
<td>85.5</td>
<td>65.7</td>
<td>63.4</td>
<td>1170</td>
<td>75.1</td>
</tr>
<tr>
<td>LaneATT [30]</td>
<td>91.7</td>
<td>76.2</td>
<td>70.8</td>
<td>50.5</td>
<td>76.3</td>
<td>86.3</td>
<td>69.5</td>
<td>64.1</td>
<td>1264</td>
<td>77.0</td>
</tr>
<tr>
<td>Proposed*</td>
<td>91.5</td>
<td>74.8</td>
<td>71.4</td>
<td>51.1</td>
<td>72.3</td>
<td>87.7</td>
<td>69.7</td>
<td>62.0</td>
<td>1507</td>
<td>76.5</td>
</tr>
<tr>
<td>Proposed</td>
<td>91.7</td>
<td>76.0</td>
<td>71.8</td>
<td>52.2</td>
<td>74.1</td>
<td>87.7</td>
<td>69.8</td>
<td>62.9</td>
<td>1509</td>
<td>77.2</td>
</tr>
</tbody>
</table>

Figure 12 shows some detection results in challenging scenarios. Although the lanes are extremely ambiguous, the proposed algorithm detects them reliably by exploiting the structural properties between the adjacent ones.

Comparison on SDLane: It is challenging to detect highly curved lanes in the anchor-based detection framework, but the proposed algorithm provides excellent results on such curved lanes. To demonstrate this, on SDLane, we compare the proposed algorithm with the state-of-the-art techniques [19, 30, 34]. LaneATT [30] is an anchor-based method considering straight lines as anchors, while RESA [34] is based on the semantic segmentation framework. Recently, CondLaneNet [19] was proposed, which yields an F-measure of 79.48% on CULane. We train these methods on SDLane using the publicly available source codes.

In Table 3, we see that the proposed algorithm is superior to the existing methods. LaneATT poorly recalls highly curved lanes, because straight anchors deviate too much from such lanes. Although RESA yields a higher recall rate, it does not detect invisible lanes reliably. CondLaneNet achieves the highest precision score, but its recall rate is still low. Figure 13 compares some detection results. LaneATT detects straight or mildly curved lanes precisely, but it fails to detect more complicated lanes, even though those lanes are visible in the images. RESA detects such complicated lanes better than LaneATT does. However, for invisible or unobvious lanes, it does not preserve the continuous lane structure in detection results. In contrast, the proposed algorithm is capable of detecting both straight and curved lanes precisely, as well as processing implicit lanes reliably. This is because the proposed algorithm generates diverse lane candidates and then localizes lanes effectively in the eigenlane space.

The proposed algorithm yields an F-measure of 80.47% on SDLane, which is significantly higher than that on the ‘Curve’ category in CULane in Table 2. This means that, with sufficiently big data, the proposed algorithm can deal with curved lanes effectively. More results are presented in the supplemental document (Section C) and video.

4.4. Ablation studies

We conduct ablation studies to analyze the efficacy of eigenlane projection and SIIC-Net components on the SDLane dataset. Also, we analyze the runtime for each component of SIIC-Net.

Efficacy of eigenlane projection: In the anchor-based detection framework, it is important to generate a set of candidates (anchors) reliably. Table 4 compares alternative methods for lane candidate generation. Method I (proposed) clusters 1,000 lane candidates in the eigenlane space of dimension $M = 6$, while II does 1,000 candidates in the original lane space of $N = 50$. III and IV, respectively, obtain 1,000 and 10,000 straight lines as done in [30]. The mean intersection over union (mIoU) scores are reported. For each lane in test data, the closest candidate in terms of IoU is found. Then, the average of these matching IoU’s is computed. Between I and II, the scores are similar. However, method I can refine lane candidates more efficiently.

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>LaneATT [30]</td>
<td>85.75</td>
<td>64.28</td>
<td>73.49</td>
</tr>
<tr>
<td>RESA [34]</td>
<td>82.35</td>
<td>72.46</td>
<td>77.99</td>
</tr>
<tr>
<td>CondLaneNet [19]</td>
<td>87.59</td>
<td>67.08</td>
<td>75.97</td>
</tr>
<tr>
<td>Proposed</td>
<td>86.04</td>
<td>75.58</td>
<td>80.47</td>
</tr>
</tbody>
</table>

Table 3. Comparison on SDLane.
Table 4. Ablation studies for the proposed eigenlane projection on the SDLane dataset. mIoU scores are reported.

<table>
<thead>
<tr>
<th></th>
<th>I (proposed)</th>
<th>II</th>
<th>III</th>
<th>IV</th>
</tr>
</thead>
<tbody>
<tr>
<td>mIoU</td>
<td>0.814</td>
<td>0.815</td>
<td>0.691</td>
<td>0.738</td>
</tr>
</tbody>
</table>

Table 5. Ablation studies for the components of SIIC-Net on the SDLane dataset.

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>I. w/o offsets</td>
<td>43.86</td>
<td>38.53</td>
<td>41.02</td>
</tr>
<tr>
<td>II. w/o heights</td>
<td>80.27</td>
<td>70.52</td>
<td>75.08</td>
</tr>
<tr>
<td>III. w/o IC+MWCS</td>
<td>82.98</td>
<td>74.82</td>
<td>78.69</td>
</tr>
<tr>
<td>IV. SIIC-Net</td>
<td>86.04</td>
<td>75.58</td>
<td>80.47</td>
</tr>
</tbody>
</table>

Using compact offset vectors (\(M < N\)), III yields poor results. By requiring 10 times more candidates, IV performs better than III does, but the gap with I is still high. This is because III and IV do not consider curved lanes. In contrast, the proposed notion of eigenlanes enables the systematic generation of curved lane candidates.

Efficacy of components in SIIC-Net: Table 5 compares several ablated methods. Method I does not use regression offsets. Method II does not use regression offsets. Method III uses the SI module and NMS only to detect road lanes. In Method III, we modify NMS as follows. First, we stop the iteration if the probability is lower than 0.5. Second, we optimize the threshold for removing redundant lanes empirically.

Compared with the full SIIC-Net in IV, Method I degrades performances severely, indicating that SIIC-Net estimates regression offsets accurately to refine detected lines. Also, from II and IV, note that the height classification improves the performance by adjusting the ending points of lanes. Last, compared with IV, III still yields lower performances in terms of all metrics even with those modifications of NMS. This means that the IC module with MWCS is required to detect lanes more precisely and more reliably.

Runtime: Table 6 shows the runtime for each stage of SIIC-Net. SI+NMS takes the longest time among the three stages, for it should perform feature pooling for all lane candidates. After the NMS process, IC+MWCS considers significantly fewer lanes, so it demands the lowest computational cost. Overall, the processing speed is about 101 frames per second, which is sufficiently fast for practical applications.

5. Conclusions

We proposed a novel algorithm to detect road lanes in the eigenlane space. First, we introduced the notion of eigenlanes, which are data-driven lane descriptors. Second, we generated a set of lane candidates by clustering training lanes in the eigenlane space. Third, we detected road lanes, by developing an anchor-based detection network SIIC-Net, from the lane candidates. Furthermore, we developed the structurally diverse dataset, containing highly curved and complicated lanes in real driving environments. Experimental results showed that the proposed algorithm provides excellent performances, especially on curved lanes.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grants funded by the Korea government (MSIT) (No. NRF-2021R1A4A1031864 and No. NRF-2022R1A2B5B03002310).
References


[14] Dongkwon Jin, Wonhui Park, Seong-Gyun Jeong, and Yinghua He, Hong Wang, and Bo Zhang. Color-based road detection in urban traffic scenes. In Intelligent Transportation Symposium, 2008. 1, 2


