Eigenlanes: Data-Driven Lane Descriptors for Structurally Diverse Lanes

Dongkwon Jin[1], Wonhui Park[1], Seong-Gyun Jeong[2], Heeyeon Kwon[2], Chang-Su Kim[1]

Korea University[1], 42dot.ai[2]

{dongkwonjin, whpark}@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,
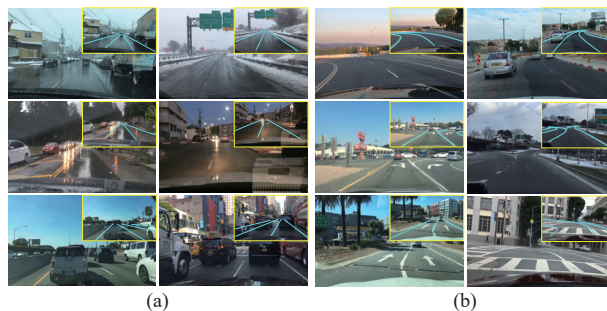

(a)   (b)

Figure 1. There are two challenging factors in lane detection. First, in (a), lanes may be implicit due to adverse weather conditions or occlusion by nearby vehicles. Second, in (b), it is difficult to design lane anchors due to the structural diversity of lanes. The ground-truth lanes are shown in cyan within the insets.

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.
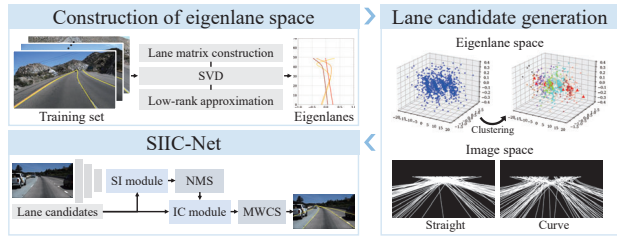

Figure 2. Overview of the proposed algorithm. It is recommended to watch the accompanying video of the proposed algorithm.

In [32], a three-branched network regresses polynomial coefficients of each lane and estimates its starting and ending points. In [25], for computational efficiency, a network selects the location of each lane on a predefined set of rows only. In [33], a unified network blends multi-scale features and combines prediction results at different levels. Qu *et al.* [26] estimated multiple keypoints and associated them to reconstruct actual lanes. Meanwhile, an anchor-based detection framework was employed for lane detection [18, 30]. These anchor-based techniques consider straight lines as lane candidates (or anchors) and generate a predefined set of candidates. Then, they classify and regress each candidate by estimating the lane probability and offset vectors. Despite providing promising results, they may fail to detect highly curved lanes. The proposed algorithm is also anchor-based but can deal with such complicated lanes successfully by employing eigenlane descriptors.

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 $\mathbf{x} = [x_1, x_2, \ldots, x_N]^\top$ be a lane, where x_i
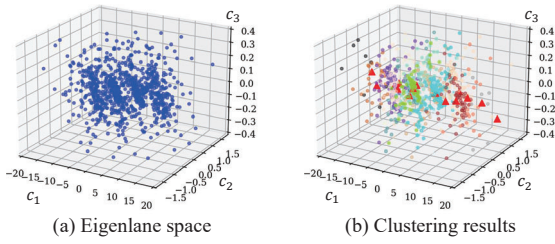
(a) Eigenlane space   (b) Clustering results

Figure 3. (a) 1,000 training lanes sampled from the TuSimple dataset [1] are visualized in the 3D eigenlane space. (b) These lanes are clustered using the K-means algorithm with $K = 16$. Each cluster is in a different color, and the centroids are depicted by red triangles.

is the x-coordinate of the ith sample and N is the number of samples. We construct a lane matrix $\mathbf{A} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_L]$ from a training set containing L lanes. Then, we apply SVD to the lane matrix $\mathbf{A}$ by

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top \qquad (1)$$

where $\mathbf{U} = [\mathbf{u}_1, \cdots, \mathbf{u}_N]$ and $\mathbf{V} = [\mathbf{v}_1, \cdots, \mathbf{v}_L]$ are orthogonal matrices and $\mathbf{\Sigma}$ 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 $\mathbf{A}$. It is known that

$$\mathbf{A}_M = [\tilde{\mathbf{x}}_1, \cdots, \tilde{\mathbf{x}}_L] = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^\top + \cdots + \sigma_M \mathbf{u}_M \mathbf{v}_M^\top \quad (2)$$

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

$$\|\mathbf{A} - \mathbf{A}_M\|_F^2 = \sum_{i=1}^{L} \|\mathbf{x}_i - \tilde{\mathbf{x}}_i\|^2 = \sum_{i=M+1}^{r} \sigma_i^2. \quad (3)$$

In (2), each approximate lane $\tilde{\mathbf{x}}_i$ is given by a linear combination of the first M left singular vectors $\mathbf{u}_1, \cdots, \mathbf{u}_M$. In other words,

$$\tilde{\mathbf{x}}_i = \mathbf{U}_M \mathbf{c}_i = [\mathbf{u}_1, \cdots, \mathbf{u}_M]\mathbf{c}_i. \qquad (4)$$

We refer to these $\mathbf{u}_1, \cdots, \mathbf{u}_M$ as *eigenlanes*, because they are eigenvectors of $\mathbf{A}\mathbf{A}^\top$. 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 $\mathbf{A}$ [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 $\{\mathbf{u}_1, \cdots, \mathbf{u}_M\}$ as the *eigenlane space*. Given a lane $\mathbf{x}$, we project it onto the eigenlane space to obtain the approximation

$$\tilde{\mathbf{x}} = \mathbf{U}_M \mathbf{c} \qquad (5)$$

where the coefficient vector $\mathbf{c}$ is given by

$$\mathbf{c} = \mathbf{U}_M^\top \mathbf{x}. \qquad (6)$$

Algorithm 1 Lane candidate generation in eigenlane space

Input: Set of training lanes $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_L\}$, $M = $ # of eigenlanes, $K = $ # of lane candidates

1: Construct the lane matrix $\mathbf{A}$ and perform SVD in (1)
2: Transform each lane $\mathbf{x}_i$ to $\mathbf{c}_i$ via (6)
3: Apply the K-means algorithm [8] to $\{\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_L\}$ to obtain K centroids $\mathbf{c}^1, \cdots, \mathbf{c}^K$.
4: Generate the lane candidate $\mathbf{l}_k = \mathbf{U}_M \mathbf{c}^k$ by inversely transforming each centroid $\mathbf{c}^k$ via (5)

Output: Set of lane candidates $\{\mathbf{l}_1, \cdots, \mathbf{l}_K\}$

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

Detection and regression in eigenlane space: Let $\{\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_L\}$ 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 $\mathbf{U}_M$ is length-preserving;

$$\|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j\| = \|\mathbf{c}_i - \mathbf{c}_j\|. \qquad (7)$$

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 $\mathbf{l} = \mathbf{U}_M \mathbf{c}$ is detected. Then, we refine it to

$$\mathbf{l} + \Delta \mathbf{l} = \mathbf{U}_M (\mathbf{c} + \Delta \mathbf{c}), \qquad (8)$$

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

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 $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \mathbf{u}_4$, which are sufficient to represent all TuSimple lanes faithfully. The first eigenlane $\mathbf{u}_1$ is a slanted line, instead of a vertical line. Most lanes are slanted from the viewpoint of a driving car. Because $\mathbf{u}_1$ achieves the best rank-1 approximation of these lanes, it is also slanted. By weighting $\mathbf{u}_1$, we can represent straight road lanes in an image. Next, $\mathbf{u}_2$ is slightly curved at the top side (far from the cameras), and $\mathbf{u}_3$ is curvier. These eigenlanes are required to represent simply curved lanes. Finally, $\mathbf{u}_4$ 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 $\mathbf{u}_1$ is slanted to
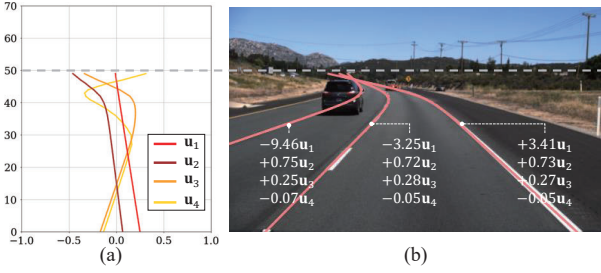
Figure 4. (a) The first four eigenlanes $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \mathbf{u}_4$ for the TuSimple dataset. (b) Three example lanes are approximated by linear combinations of the four eigenlanes.
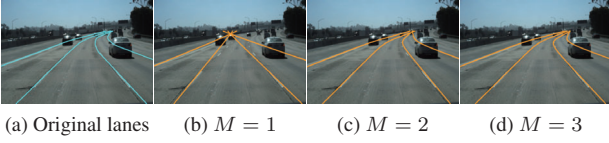

(a) Original lanes   (b) $M = 1$ (c) $M = 2$ (d) $M = 3$

Figure 5. Rank-M approximation: original lanes in (a) are reconstructed by the first (b) one, (c) two, and (d) three eigenlanes, respectively.

the left. Thus, the coefficients for $\mathbf{u}_1$ for these two lanes are negative. For all lanes, the coefficients for $\mathbf{u}_2$ and $\mathbf{u}_3$ 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 $\mathbf{u}_1$, 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 $\{\mathbf{l}_1, \cdots, \mathbf{l}_K\}$, 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 en-

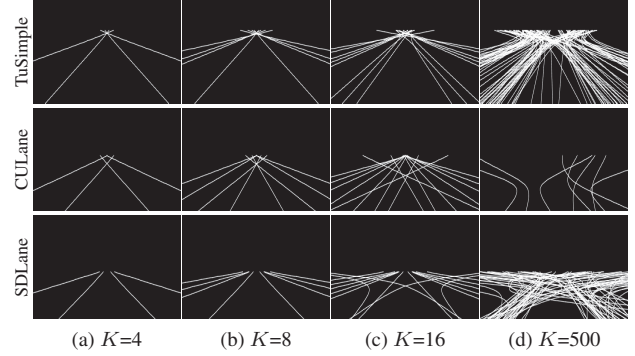(a) K=4   (b) K=8   (c) K=16   (d) K=500

Figure 6. Lane candidates, generated by the K-means clustering, for the TuSimple [1], CULane [22], and SDLane datasets. In (d), only curved lanes are shown among 500 candidates. In CULane, there are only 8 curved lanes among those 500 candidates.

coder 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 $X_{\mathrm{a}} = [X_{\mathrm{a}}^1, X_{\mathrm{a}}^2, \ldots, X_{\mathrm{a}}^{C_1}] \in \mathbb{R}^{H \times W \times C_1}$ 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_{a} using convolutional layers to yield $X_{\mathrm{s}} \in \mathbb{R}^{H \times W \times C_2}$. The decoder processes X_{s} 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 $\mathbf{l}_k$, 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 $Y_{\mathrm{s}} = [Y_{\mathrm{s}}^1, Y_{\mathrm{s}}^2, \ldots, Y_{\mathrm{s}}^{C_2}] \in \mathbb{R}^{K \times C_2}$ by averaging the features of pixels along $\mathbf{l}_k$;

$$Y_{\mathrm{s}}^c(k) = \tfrac{1}{|\mathbf{l}_k|} \sum_{\mathbf{p} \in \mathbf{l}_k} X_{\mathrm{s}}^c(\mathbf{p}) \qquad (9)$$

for $1 \leq k \leq K$ and $1 \leq c \leq C_2$, where $|\mathbf{l}_k|$ denotes the number of pixels in $\mathbf{l}_k$. Then, two probability vectors and a lane offset matrix are obtained by

$$P = \sigma(f_1(Y)), \quad H = \sigma(f_2(Y)), \quad O = f_3(Y) \qquad (10)$$

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 $\mathbf{l}_k$, 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 $O_k = \Delta\mathbf{c}_k$ 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
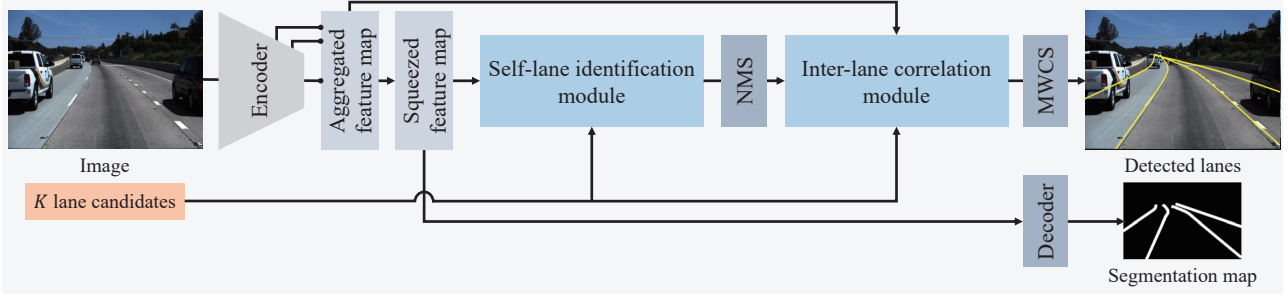
17166

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.
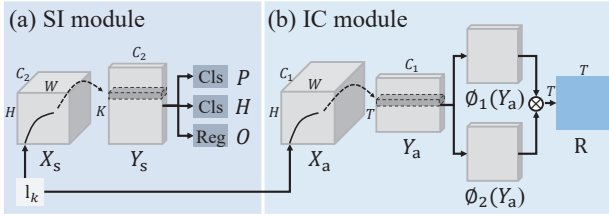

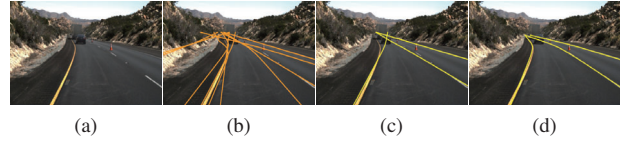Figure 8. Block diagrams of the SI and IC modules.

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.

an NMS process after the SI module. We select the most reliable lane $\mathbf{l}_{i^\star}$ by

$$i^\star = \arg\max_i P_i. \qquad (11)$$

Then, we remove overlapping lanes, whose the intersection over union (IoU) ratios with the selected lane are higher than a threshold. We perform this process T times to select the T reliable lanes. The default T is 10. Note that we focus on reducing false negatives, rather than false positives. Figure 9(b) shows 10 selected lanes after the NMS process.

Inter-lane correlation (IC) module: In general, adjacent lanes are equally distanced in road environments. Also, under the perspective projection, lanes intersect at a vanishing point in a 2D image. Because of these structural constraints, lanes are highly correlated with one another. To exploit this correlation, we design the IC module, which estimates the relation score between every pair of selected lanes. Figure 8(b) shows the structure of the IC module.

Given the aggregated feature map X_a, IC yields a lane feature map Y_a similarly to (9). Y_a is a $T \times C_1$ matrix, in which each row contains the C_1-dimensional feature vector for a selected lane. Then, it obtains the relation matrix

$$\mathbf{R} = \phi_1(Y_\mathrm{a}) \times \phi_2(Y_\mathrm{a})^\top \qquad (12)$$

of size $T \times T$. Here, ϕ_1 and ϕ_2 are feature transforms, implemented by convolution layers and the l_2-normalization. Thus, each element of $\mathbf{R}$ is a score in $[-1, 1]$, representing how compatible the corresponding pair of lanes are.

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

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

$$E_{\mathrm{compatible}}(\theta) = \sum_{i \in \theta} \sum_{j \in \theta, j > i} w(v_i, v_j). \qquad (13)$$

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

$$\theta^\star = \arg\max_\theta E_{\mathrm{compatible}}(\theta) \qquad (14)$$

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

Next, we refine each lane in the optimal clique $\theta^\star$ by $\mathbf{U}(\mathbf{c}_{v_i} + \Delta\mathbf{c}_{v_i})$, where $\Delta\mathbf{c}_{v_i}$ is the offset vector in the eigenlane 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.
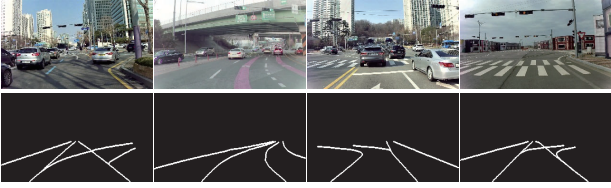
17167

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.

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

$$\text{FPR} = \frac{F_{\text{pred}}}{N_{\text{pred}}}, \quad \text{FNR} = \frac{M_{\text{pred}}}{N_{\text{gt}}} \qquad (15)$$

where F_{pred} is the number of incorrectly predicted lanes, N_{pred} is that of predicted lanes, M_{pred} is that of missed lanes, and N_{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

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}}, \quad \text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \qquad (16)$$

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

	Accuracy	FPR	FNR
LaneNet [21]	96.38	0.0780	0.0244
SCNN [22]	96.53	0.0617	**0.0180**
SAD [13]	<u>96.64</u>	0.0602	<u>0.0205</u>
UFast [25]	95.82	0.1905	0.0392
RESA [34]	**96.82**	0.0363	0.0248
LaneATT [30]	95.63	<u>0.0353</u>	0.0292
Proposed	95.62	**0.0320**	0.0399

Figure 11. Detection results of the proposed algorithm on the TuSimple dataset. Detected lanes are depicted in green, while false negatives are in red.

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

$$\text{F-measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \qquad (17)$$

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 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.

Category	Normal	Crowded	Night	No line	Shadow	Arrow	Dazzle	Curve	Crossroad	Total
SCNN [22]	90.6	69.7	66.1	43.4	66.9	84.1	58.5	65.7	1990	71.6
SAD [13]	90.7	70.0	66.3	43.5	67.0	84.4	59.9	65.7	2052	71.8
UFast [25]	90.7	70.2	66.7	44.4	69.3	85.7	59.5	69.7	2037	72.3
Curve-Nas [33]	90.7	72.3	68.9	49.4	70.1	85.8	67.7	68.4	1746	74.8
RESA [34]	92.1	73.1	69.9	47.7	72.8	88.3	69.2	70.3	1503	75.3
LaneATT* [30]	91.1	73.0	69.0	48.4	70.9	85.5	65.7	63.4	1170	75.1
LaneATT [30]	91.7	76.2	70.8	50.5	76.3	86.3	69.5	64.1	1264	77.0
Proposed*	91.5	74.8	71.4	51.1	72.3	87.7	69.7	62.0	1507	76.5
Proposed	91.7	76.0	71.8	52.2	74.1	87.7	69.8	62.9	1509	77.2

Figure 12. Detection results of the proposed algorithm on five challenging categories in the CULane dataset.

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, Cond-LaneNet [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. Cond-LaneNet 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

Table 3. Comparison on SDLane.

	Precision	Recall	F-measure
LaneATT [30]	85.78	64.28	73.49
RESA [34]	82.35	72.46	77.09
CondLaneNet [19]	87.59	67.08	75.97
Proposed	86.04	75.58	80.47

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 SD-Lane 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
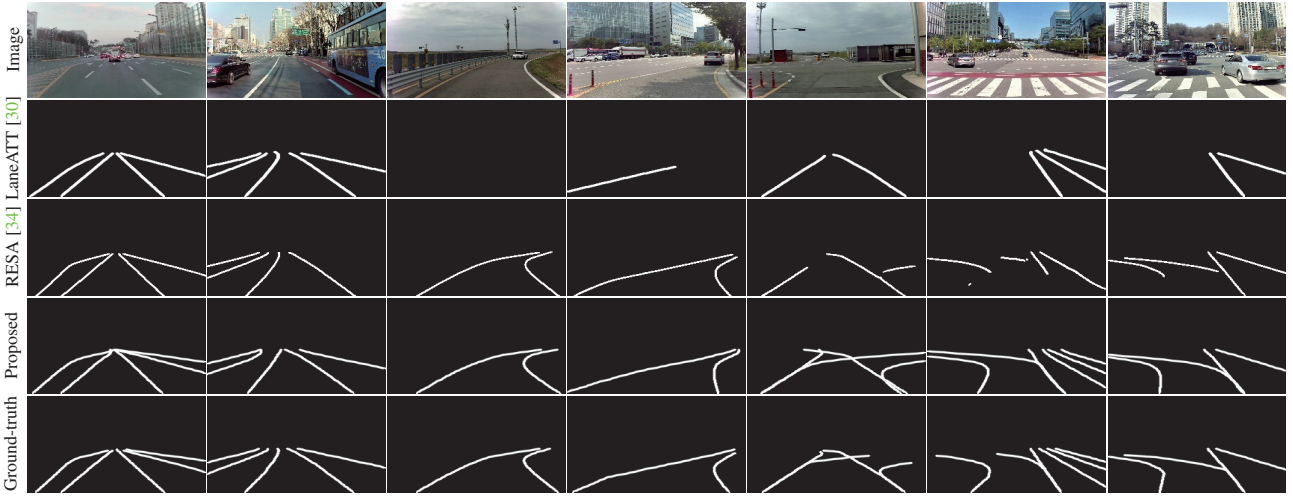
Figure 13. Comparison of lane detection results on the SDLane dataset.

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

	I (proposed)	II	III	IV
mIoU	0.814	0.815	0.691	0.738

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

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

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. In Method II, height classification results are not used. 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

Table 6. Analysis of running times of the proposed SIIC-Net. The processing times in seconds per frame are reported.

Encoding	SI+NMS	IC+MWCS	Total
0.0036s	0.0042s	0.0021s	0.0099s

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

References

[1] TuSimple benchmark. https://github.com/TuSimple/tusimple-benchmark. 2, 3, 4, 6

[2] Mohamed Aly. Real time detection of lane markers in urban streets. In *Intelligent Vehicles Symposium*, 2008. 1, 2

[3] Richard H. Bartels, John C. Beatty, and Brian A. Barsky. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann, 1995. 2

[4] Avrim Blum, John Hopcroft, and Ravindran Kannan. Foundations of Data Science. 2015. 1, 2, 3

[5] Mohsen Ghafoorian, Cedric Nugteren, Nóra Baka, Olaf Booij, and Michael Hofmann. EL-GAN: Embedding loss driven generative adversarial networks for lane detection. In *ECCV Workshops*, 2018. 1, 2

[6] William J. Gordon and Richard F. Riesenfeld. B-spline curves and surfaces. *Computer Aided Geometric Design*, pages 95–126, 1974. 2

[7] Qi Han, Kai Zhao, Jun Xu, and Ming-Ming Cheng. Deep Hough transform for semantic line detection. In *Proc. ECCV*, 2020. 4

[8] J. A. Hartigan and M. A. Wong. A K-means clustering algorithm. *Journal of the Royal Statistical Society, Series C (Applied Statistics)*, 28(1):100–108, 1979. 3

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE CVPR*, 2016. 4

[10] Yinghua He, Hong Wang, and Bo Zhang. Color-based road detection in urban traffic scenes. *IEEE Trans. Intelligent Transportation Systems*, 5(4):309–318, 2004. 1, 2

[11] Aharon Bar Hillel, Ronen Lerner, Dan Levi, and Guy Raz. Recent progresss in road and lane detection: A survey. *Mach. Vis. Appl.*, 25(3):727–745, 2014. 1, 2

[12] Yuenan Hou, Zheng Ma, Chunxiao Liu, Tak-Wai Hui, and Chen Change Loy. Inter-region affinity distillation for road marking segmentation. In *Proc. IEEE CVPR*, 2020. 1, 2

[13] Yuenan Hou, Zheng Ma, Chunxiao Liu, and Chen Change Loy. Learning lightweight lane detection CNNs by self attention distillation. In *Proc. IEEE ICCV*, 2019. 1, 2, 6, 7

[14] Dongkwon Jin, Wonhui Park, Seong-Gyun Jeong, and Chang-Su Kim. Harmonious semantic line detection via maximal weight clique selection. In *Proc. IEEE CVPR*, 2021. 5

[15] Hei Law and Jia Deng. CornerNet: Detecting objects as paired keypoints. In *Proc. ECCV*, 2018. 1

[16] David C. Lay. *Linear Algebra and Its Applications*. Pearson, 2007. 2, 3

[17] Jun-Tae Lee, Han-Ul Kim, Chul Lee, and Chang-Su Kim. Semantic line detection and its applications. In *Proc. IEEE ICCV*, 2017. 4

[18] Xiang Li, Jun Li, Xiaolin Hu, and Jian Yang. Line-CNN: End-to-end traffic line detection with line proposal unit. *IEEE Trans. Intelligent Transportation Systems*, 21(1):248–258, 2019. 1, 2, 6

[19] Lizhe Liu, Xiaohao Chen, Siyu Zhu, and Ping Tan. Cond-LaneNet: A top-to-down lane detection framework based on conditional convolution. In *Proc. IEEE ICCV*, 2021. 7

[20] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot multibox detector. In *Proc. ECCV*, 2016. 1

[21] Davy Neven, Bert De Brabandere, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Towards end-to-end lane detection: An instance segmentation approach. In *Intelligent Vehicles Symposium*, 2018. 1, 2, 6

[22] Xingang Pan, Jianping Shi, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Spatial as deep: Spatial CNN for traffic scene understanding. In *Proc. AAAI*, 2018. 1, 2, 4, 6, 7

[23] Wonhui Park, Dongkwon Jin, and Chang-Su Kim. Eigen-contours: Novel contour descriptors based on low-rank approximation. In *Proc. IEEE CVPR*, 2022. 2

[24] Jonah Philion. FastDraw: Addressing the long tail of lane detection by adapting a sequential prediction network. In *Proc. IEEE CVPR*, 2019. 2

[25] Zequn Qin, Huanyu Wang, and Xi Li. Ultra fast structure-aware deep lane detection. In *Proc. ECCV*, 2020. 2, 4, 6, 7

[26] 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 CVPR*, 2021. 2

[27] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proc. IEEE CVPR*, 2016. 1

[28] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Proc. NIPS*, 2015. 1

[29] Abraham Savitzky and Marcel JE. Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry*, 36(8):1627–1639, 1964. 2

[30] 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 CVPR*, 2021. 1, 2, 6, 7, 8

[31] Matthew A. Turk and Alex P. Pentland. Face recognition using eigenfaces. In *Proc. IEEE CVPR*, 1991. 2

[32] Bingke Wang, Zilei Wang, and Yixin Zhang. Polynomial regression network for variable-number lane detection. In *Proc. ECCV*, 2020. 1, 2

[33] Hang Xu, Shaoju Wang, Xinyue Cai, Wei Zhang, Xiaodan Liang, and Zhenguo Li. CurveLane-NAS: Unifying lane-sensitive architecture search and adaptive point blending. In *Proc. ECCV*, 2020. 2, 7

[34] 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. AAAI*, 2021. 1, 2, 6, 7, 8

[35] Shengyan Zhou, Yanhua Jiang, Junqiang Xi, Jianwei Gong, Guangming Xiong, and Huiyan Chen. A novel lane detection based on geometrical model and gabor filter. In *Intelligent Vehicles Symposium*, 2010. 1, 2