LaneAF: Robust Multi-Lane Detection with Affinity Fields

Hala Abualsaud†, Sean Liu†, David Lu†, Kenny Situ†, Akshay Rangesh†, and Mohan M. Trivedi
Laboratory for Intelligent & Safe Automobiles, UC San Diego
{habualsa, selll18, dblu, ksitu, arangesh, mtrivedi}@ucsd.edu

Abstract—This paper presents an approach to lane detection via lane instance segmentation by training models to predict binary segmentation masks and affinity fields. These affinity fields, along with the binary masks, can then be used to cluster lane pixels horizontally and vertically into corresponding lane instances as a post-processing step. This clustering is achieved through a simple row-by-row decoding process with little overhead; such an approach allows LaneAF to detect a variable number of lanes without assuming a fixed or maximum number of lanes. Moreover, this form of clustering is more interpretable in comparison to previous visual clustering approaches, and can be analyzed to identify and correct sources of error. Qualitative and quantitative results presented on the popular TuSimple benchmark demonstrate the model’s ability to detect and cluster lanes effectively and robustly.

I. INTRODUCTION

LANE DETECTION is the process of automatically perceiving the shape and position of marked lanes on the road preceding a vehicle and is a crucial component of autonomous driving systems, directly influencing the guidance and steering of vehicles while also aiding the interaction between numerous agents on the road. As the number of drivers on the roads has increased, autonomous driving systems have received considerable attention in the automotive and tech industries as well as in academia [1]. According to the Insurance Institute for Highway Safety (IIHS), in the US alone, car accidents claimed 36,560 lives in 2018, underscoring the importance of any technology that can help prevent crashes.

Since roads commonly have different types of lane lines (solid white, broken white, solid yellow, etc.), each of which have specific implications with regards to how vehicles may interact with them, automated lane detection systems can also help alert drivers when there are changes in lane topology on the road. Furthermore, there are several factors that make lane detection a challenging task. Firstly, there is a wide variety of road infrastructure in use around the world. Additionally, the lane detection system must be able to identify instances where lanes are ending, merging, and splitting. Finally, the lane detection system must possess the ability to discern worn or unclear lane markings. Precise detection of lanes would enable more robust trajectory prediction of surrounding vehicles; as discussed in [2], this is critical for successful path planning in autonomous driving. Therefore, while lane detection is a significant and complex task, it is a key factor in developing any autonomous vehicle system.

In this paper, lane detection is approached using Deep Layer Aggregation (DLA) as the backbone network because of its accuracy and efficiency; in [3], the authors used DLA to achieve best-in-class accuracy on semantic segmentation of the CityScapes dataset. Since CityScapes is a large-scale and challenging dataset with 19 semantic categories, it was believed that DLA could achieve or surpass current state-of-the-art performance in the lane detection task, which involves significantly fewer categories.

While binary classification is used for the detection of lanes in LaneAF, a limitation of this type of classification is that it uses a single-channel output, which does not allow for the identification of separate lane entities. To perform instance segmentation, we developed a novel technique using affinity fields (see Figure 1). Affinity fields were introduced in [4] for the purpose of multi-person 2D pose estimation, and are unit vectors that encode location and orientation. This technique was also used for the detection of hands inside a vehicle, as demonstrated in [5]. In this paper, we have

†authors contributed equally
Code: https://github.com/sell18/LaneAF
The authors are with the Laboratory for Intelligent and Safe Automobiles at the University of California, San Diego, CA 92092, USA.
email: {habualsa, selll18, dblu, ksitu, arangesh, mtrivedi}@ucsd.edu

Fig. 1: The standard approach to lane detection treats each lane as a separate class and trains a model to perform multi-class segmentation. In our approach, we instead propose to train models that output binary segmentation masks and affinity fields, which can then be decoded together to produce multiple lane instances.
defined two types of affinity fields, the horizontal affinity field (HAF) and vertical affinity field (VAF). A vector in the VAF encodes not only the location of a given lane pixel, but also the direction in which the next lane pixel above it is located. An HAF vector points toward the center of a pixel row that constitute the width of a given lane line.

It is these affinity fields that enable unique lane instances to be identified and segmented. Since these affinity fields are present wherever the binary classification has identified lane pixels, they are not bound to a pre-determined number of lanes. The model is therefore agnostic to the number of lanes present in a given dataset. The main contributions of this paper are as follows:

1) We propose affinity fields that are suitable for clustering/associating pixels belonging to amorphous entities like lanes.
2) A procedure to train models that predict binary segmentation masks and affinity fields for the purpose of lane instance segmentation.
3) We introduce efficient methods for generating and decoding such affinity fields into an unknown number of clustered lane instances.

II. RELATED RESEARCH

Lane detection has traditionally been tackled by feature-based approaches which then evolved to model-based approaches to detect lane boundaries. However, these are not practical in real world scenarios since they require the road scene to be perfect in order to work effectively. One of the past approaches that aimed to enhance lane estimation was to use contextual cues to improve computational efficiency and accuracy of lane detection [6]. Currently, data-driven approaches are commonly used to detect both lane boundaries as well as lane regions.

While several shortcomings of the traditional lane detection methods (i.e. lane segmentation via hand-crafted features) have been resolved with more robust methods in recent years, there is still room for improvement. Deep learning has provided a solution to many of these issues, with neural networks being used to deal with lane detection as a semantic segmentation problem.

Furthermore, deep learning helped to introduce a new era of lane detection in recent years; it proved an obvious improvement in the robustness of the lane detection problem. A lot of new approaches use deep learning to tackle lane detection such as in [7], [8], [9], [10]. One of the highlighted approaches of lane detection in recent years [7] utilizes a spatial convolutional neural network (CNN). This method has improved performance over conventional CNN methods since it provides spatial information by computing slice by slice convolution in feature maps, enabling information to be transferred between pixels within each layer. It achieved an accuracy of 96.53% with the TuSimple dataset. Other approaches that have used CNN is proposed in [8] where the authors have also combined a recurrent neural network (RNN) with a CNN for lane prediction.

Other methods that have also been investigated in recent years were emphasizing improving the loss functions such as in [11] and [12]. In [11], the authors introduced Self Attention Distillation (SAD) loss to avoid models that propagate data sequentially and to decrease inference time, which is essential for real-time tasks such as lane detection. However, the fully connected layer that the SAD model employs is computationally expensive and cannot adapt to any number of lanes.

Some approaches tried to improve the network architecture as in [7], [13], [14], [15], and [16]. In [13], the authors developed 3D-LaneNet, a network that predicts the 3D layout of lanes using a single image. In [14], a sequential prediction network has been used to avoid heuristic-based clustering post-processing. Another network architecture was presented in [15] with two elements: a deep network which generates weighted pixel coordinates in addition to a differentiable weighted least-squares fitting module. Generative adversarial networks (GANs) using an embedding loss were used in [9] to better preserve the structure of lanes and to mitigate the problem of complex post processing; 96% accuracy on the TuSimple dataset was obtained. [16] uses a combination of LiDAR and camera sensors for their network in order to obtain accurate results of lane detection in 3D space directly.

Other approaches pay more attention to real-time lane detection improvement such as in [17], [18], [19] and [20]. In [17], a spatio-temporal deep learning method was proposed to mitigate the errors that can occur when experiencing harsh weather or other complex problems in the road, jeopardizing the accuracy of detecting a lane in the scene. Meanwhile, in [18], lane markers were tracked temporally, and in [20], a combination of instance segmentation and classification was used as an end-to-end deep learning real-time method to avoid reliance on two-step detection networks.

Moreover, [21] presents a method to deal with lane detection as an instance segmentation problem so that each lane can be trained in an end-to-end manner, coping with changing lane numbers on the road.

Although recent methods of lane detection show high accuracy when applied to the popular published datasets, some of the drawbacks of these current methods are that they are not robust enough in terms of experiencing occlusion and that they require a preset number of lanes in a scene in order to actually work. Thus, they cannot work for any random number of lanes in a road scene.

The methods in this paper were inspired by [4], which presented an approach to 2D pose estimation of multiple people in an image through Part Affinity Fields (PAFs). The technique introduced takes an image as input and passes it to a two-branch CNN, obtains the confidence maps to detect body parts, utilizes PAFs for parts association, parses using a greedy algorithm, and finally assembles them into the final image with estimated poses. The main takeaway from [4] is that parsing based on PAFs adds robustness with regards to part detection and association.
Algorithm 1 Generating affinity fields

Inputs:

\[ \text{SEG}(H \times W) : \text{ground truth segmentation} \]
\[ l_{\text{max}} : \text{maximum number of lanes} \]

\[ \text{HAF}, \text{VAF} \leftarrow \text{zeros}(H, W, 2) \quad \text{▷ initialize affinity fields} \]

for \( l \leftarrow 1 \) to \( l_{\text{max}} \) do
  \( \text{prev_cols} \leftarrow \text{find}(\text{SEG}[\text{row}, H] == l) \) \quad \text{▷ initialize}
  for \( \text{row} \leftarrow H - 1 \) to \( 1 \) do
    \( \text{cols} \leftarrow \text{find}(\text{SEG}[\text{row}, :] == l) \) \quad \text{▷ find lane pixels}
    \( \text{HAF}\{	ext{row}, \text{col}, 0\} \leftarrow 1.0 \quad \text{▷ points to the right} \)
    else if \( \text{col} > \text{mean}(	ext{cols}) \) then
      \( \text{HAF}\{	ext{row}, \text{col}, 0\} \leftarrow -1.0 \quad \text{▷ points to the left} \)
    else
      \( \text{HAF}\{	ext{row}, \text{col}, 0\} \leftarrow 0.0 \)
  end for
  end for
/* vertical affinity fields */

for \( \text{col} \) in \( \text{prev_cols} \) do
  \( \text{v} \leftarrow [1, \text{mean}(	ext{cols}) - \text{col}] \quad \text{▷ points to the mean} \)
  \( \text{v}_\text{norm} \leftarrow \text{v} / ||\text{v}|| \quad \text{▷ unit vector} \)
  \( \text{VAF}\{\text{row} + 1, \text{col}, 0\} \leftarrow \text{v}[0] \)
  \( \text{VAF}\{\text{row} + 1, \text{col}, 1\} \leftarrow \text{v}[1] \)
end for

\( \text{prev_cols} \leftarrow \text{cols} \)
end for

return \( \text{HAF}, \text{VAF} \)

Algorithm 2 Decoding affinity fields

Inputs:

\[ \text{BW}(H \times W) : \text{binary segmentation mask} \]
\[ \text{HAF}(H \times W \times 2) : \text{horizontal affinity field} \]
\[ \text{VAF}(H \times W \times 2) : \text{vertical affinity field} \]
\( \tau : \text{clustering threshold} \)

\[ \text{SEG} \leftarrow \text{zeros}(H, W) \quad \text{▷ initialize segmentation output} \]

\( \text{lane_end_points} \leftarrow \text{[]} \)

\( \text{for} \ \text{row} \leftarrow H - 1 \) to \( 1 \) do
  \( \text{cols} \leftarrow \text{find}(\text{BW}[\text{row}, :] > 0) \) \quad \text{▷ find foreground pixels}
  \( \text{for} \ \text{col} \) in \( \text{cols} \) do
    \( \text{cur_dir} \leftarrow \text{HAF}[\text{row}, \text{col}, 0] \)
    \( \text{if} \ \text{cur_dir} > 0 \) and \( \text{prev_dir} < 0 \) then
      \( \text{end cluster} \)
      \( \text{clusters}.\text{append}(\text{cluster}) \)
    else
      \( \text{continue cluster} \)
      \( \text{cluster}.\text{append}(\text{col}) \)
  end for
  \( \text{for} \ (\text{lane_id}, \text{points}) \) in \( \text{lane_end_points} \) do
    \( \text{pred_points} \leftarrow \text{add_vaf}\{\text{points}, \text{VAF}\} \)
    \( \text{▷ add VAF vectors to points} \)
    \( \text{cluster}, e \leftarrow \text{find closest}(\text{pred_points}, \text{clusters}) \)
    \( \text{▷ find cluster closest to predicted points, and corresponding error} \)
    \( \text{if} \ e > \tau \) then
      \( \text{continue} \)
    end if
    \( \text{SEG}[\text{row}, \text{cluster}] = \text{lane_id} \)
  end for
end for

\( \text{lane_end_points}.\text{update}([\text{cluster}]) \)
  \( \text{▷ update latest points added to lane} \)

\( \text{for} \ \text{cluster} \) in \( \text{cluster} \) do
  \( \text{if} \ \text{cluster} \) is not assigned then
    \( \text{lane_end_points}.\text{append}([\text{cluster}]) \)
    \( \text{▷ spawn new lane} \)
  end if
end for

return \( \text{SEG} \)

III. METHODOLOGY

A. Data Pre-processing

In the TuSimple dataset, we are given training video clips with annotated frames for both training and testing. To train with these annotated frame images, we first created the ground truth labels for each image. Our dataloader parses the ground truth json files to create the segmented labels for each image. A binary heatmap was also added with 1 representing a lane pixel and 0 representing a background pixel to train the binary segmentation branch.

Another factor we considered was the input and output parameters of our backbone architecture, DLA-34. DLA-34 requires a specific input size of 1280 \( \times \) 768 image for the input and downsizes our image by a factor of 4 as an output, since it is a fully convolutional network that does not upsample to the original size. To account for these factors, we rescaled the input images to the desired 1280 \( \times \) 768 image size dynamically during run-time with PyTorch transformations and also reshaped the output ground truth segmented labels and heatmaps to 320 \( \times \) 192, which is exactly a quarter of the input size. Therefore, with the output generated from DLA-34, we are able to calculate the loss, propagate the error back through the network, and update the parameters.

Lastly, we normalized the input images to (0.485, 0.456, 0.406) and (0.229, 0.224, 0.225), which are the normalized values for the RGB means and standard deviations, respectively, for ImageNet. This was implemented because our DLA-34 model has ImageNet pre-trained weights. These normalized values for our inputs allow for a quicker convergence of our model since computing resources would not need to be expended on calculating the means and standard deviation of the data.

B. Deep Layer Aggregation

While conventional deep learning models such as convolutional deep network approaches are increasingly being applied as the backbone architecture to several tasks over the last decade, aggregation dimension (i.e. width and depth) has not been fully explored in all tasks. The DLA model has
optimum parameter numbers; thus, even when using the DLA network with fewer parameters, which means a more efficient model and less time training the network, we can still achieve high accuracy. The structure of the DLA-34 model we used as represented in [3] contains two structures for deep layer aggregation: iterative deep aggregation (IDA) and hierarchical deep aggregation (HDA). IDA aims to integrate scales and resolution while HDA aims to integrate features from all modules and channels.

Aggregation starts at the smallest and shallowest scale; it then merges larger and deeper scales in an iterative method. This procedure allows shallow features to be refined as they are propagated through various aggregation stages. The equation below represents the iterative deep aggregation function for layers \(x_1\) through \(x_n\), where \(x\) represents deeper and more semantic information as \(N\), the aggregation node, increases:

\[
I(x_1, ..., x_n) = \begin{cases} 
  x_1, & \text{if } n = 1 \\
  I(N(x_1, x_2), ..., x_n), & \text{otherwise}
\end{cases}
\]

Hierarchical deep aggregation integrates stages as well as blocks in a tree structure to maintain and integrate feature channels. HDA merges shallower and deeper layers together to learn deeper combinations that express more of the feature hierarchy [3].

C. Affinity Fields

We implemented a parallel branch to the binary segmentation branch that predicts the Horizontal Affinity Field (HAF) and Vertical Affinity Field (VAF) for instance segmentation of different lane markings. Specifically, affinity fields are unit vectors that point towards the object of interest. In lane detection, for the VAF, this consists of each lane pixel pointing toward the mean location of the next row’s lane pixels. For the HAF, this consists of the lane pixels comprising the width of the identified lane line pointing toward the center of that lane line. Our approach can be summarized in the following two steps:

1) Generating VAFs and HAFs: based on the ground truth label of the images, we generated ground truth VAFs and HAFs for the entire dataset as in Algorithm 1. We take in the image size and produce channels corresponding to \(x\) and \(y\) components of unit vectors. Figure 2 depicts conceptual visualization of the VAFs and HAFs.

2) Decoding: to identify and cluster the foreground pixels into unique lane instances, the affinity field decoder takes the affinity fields and binary segmented image as inputs, as shown in Figure 3, and follows the procedure specified in Algorithm 2.

With these two steps, we use an \(L2\) norm (same as in [4]) to compute the losses between our predicted affinity fields and the ground truth affinity fields that were generated based on the ground truth label of the images. For LaneAF, we combine the DLA backbone with our affinity fields to predict the segmented and uniquely identified lanes. Our entire model architecture is displayed in Figure 3.

D. Losses

For our binary segmentation branch of DLA-34, weighted binary cross-entropy loss, a standard loss metric in imbalanced binary segmentation tasks, was used. It first implements a sigmoid activation to the outputs of our model, which normalizes the logit values of the output to a standardized 0-1 output representing the probability of the pixel being a lane. It is then calculated using the normal cross entropy loss, defined as:

\[
L_{BCE} = -w_i[log(y_i) + (1 - t_i)log(1 - y_i)]
\]

where \(t_i\) is the ground truth value for the pixel and \(y_i\) is the sigmoid output of that pixel value. Since this is an unbalanced segmentation task, a weight \(w\) of 9.6 was used to increase penalization for an incorrect lane pixel label because it appears less frequently in the training images versus non-lane pixels. To further account for the imbalanced dataset, an intersection over union loss was added and is defined as:

\[
L_{IoU} = 1 - \frac{t_iy_i}{t_i + y_i - t_iy_i}
\]

For the affinity field branches of the model, an \(L1\) regression loss was applied to the foreground pixels of both the vertical and horizontal affinity fields, which were summed as below:

\[
L_{AF} = |t_i - y_i|_{VAF} + |t_i - y_i|_{HAF}
\]

The total loss is the sum of the weighted binary cross-entropy, intersection over union, and affinity field losses:

\[
L_{total} = L_{BCE} + L_{IoU} + L_{AF}
\]
Fig. 3: LaneAF architecture: after an input image has passed through DLA-34, three separate outputs are generated: a binary segmented image, Vertical Affinity Fields (VAFs), and Horizontal Affinity Fields (HAFs), all of which are combined in our decoder to produce instance segmentation results.

IV. EXPERIMENTAL EVALUATION

A. Dataset

The TuSimple dataset from the TuSimple Lane Detection Challenge was selected for training and testing of the network. This dataset contains 3,626 training video clips with annotated frames and 2,782 video clips for testing. Each video consists of a one-second-long clip comprised of 20 individual frames. The annotations are provided in json format and indicate the vehicle’s current (ego) lane and the left/right lanes. The dataset features good and fair weather conditions with various daytime lighting and traffic conditions. It also employs highway roads with differing numbers of lanes, ranging from two lanes up to four lanes.

### TABLE I: Characteristics of Lane Detection Datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>TuSimple</th>
<th>CULane</th>
</tr>
</thead>
<tbody>
<tr>
<td># Frames</td>
<td>6,408</td>
<td>133,325</td>
</tr>
<tr>
<td>Train</td>
<td>3,268</td>
<td>88,880</td>
</tr>
<tr>
<td>Validation</td>
<td>358</td>
<td>9,675</td>
</tr>
<tr>
<td>Test</td>
<td>2,782</td>
<td>34,680</td>
</tr>
<tr>
<td>Resolution</td>
<td>1280 × 720</td>
<td>1640 × 590</td>
</tr>
</tbody>
</table>

Meanwhile, the CULane Dataset from [7] contains 88,880 training video clips with annotated frames and 34,680 video clips for testing. This dataset also contains many challenging traffic scenarios with much more data than TuSimple. A summary of both of the data is compiled in Table I.

B. Metrics

We used the same evaluation metrics used in past literature to make a representative comparison between our model and prior work. This consisted of the official metric of the TuSimple dataset (accuracy), the false positive (FP) rate, and the false negative (FN) rate. The TuSimple accuracy is calculated as:

\[
\text{Accuracy} = \frac{N_{\text{pred}}}{N_{\text{gt}}}
\]

where \(N_{\text{pred}}\) is the number of lane points that have been correctly predicted and \(N_{\text{gt}}\) is the number of ground-truth lane points.

C. Results

The experimental parameters and training settings that were used are listed below:

- Number of epochs = 60
- Batch size = 2
- Adam optimizer with Learning Rate = 0.0001 and Weight Decay = 0.001

Performance results from our model on the TuSimple test set are shown in Table II; it can be seen that we have achieved comparable lane detection segmentation accuracy with other state-of-the-art lane detection models. While we obtained superior results when compared to other backbone architectures such as ResNet-18, ResNet-34, or ENet, without additional post-processing, our approach falls slightly short of current state-of-the-art models such as SCNN, LaneNet and ENet-SAD in terms of accuracy. With that said, our false positive rate sets a new standard, indicating that the model does not falsely detect a lane pixel as often as other models and that LaneAF’s multiple branch approach leads to
converge. Successful discrimination of lane instances even as the lanes instance based on affinity field decoding. Of note is the coded affinity fields; each color represents a unique lane instance. The Vertical Affinity Fields point along the lane towards the mean location of the next row’s lane pixels, which is based on the Horizontal Affinity Field outputs. This is visualized in the yellow box of Figure 4b, where for each unique lane instance, the unit vector points towards the next row’s mean lane pixel location. For both Figures 4a and 4b, the blue boxes clearly display how the Vertical Affinity Fields and Horizontal Affinity Fields are implemented on a single detected lane instance.

In Figure 5, we can see the strengths and weaknesses in our approach. In the top-right output, a simple two-lane road with three lane instances has been successfully detected, different from our previous three-lane road with four lane instances, showing that our approach is not limited to a set number of lanes. This is further seen in the top-left output, where LaneAF detects multiple curved lane instances as well as an on-ramp on the right side, a valid lane instance that was not even considered a lane pixel in the ground truth mask, exemplifying the robustness of our method. In the bottom-right output, the model identifies splitting lanes due to the coming off-ramp, again showing our method is not constrained to lanes adjacent to the ego vehicle. Finally, in the bottom-left output, LaneAF successfully merges lane instances, but also erroneously identifies a lane line in the sky due to the contour produced from a contrail.

Qualitative results can be seen in Figures 4a and 4b. The clustered outputs shown here were created using the affinity field decoder, outlined in Algorithm 2. In the detailed regions to the right of both figures, it is clear that the affinity fields point towards the mean of a found lane cluster, working off of the binary segmentation output. In Figure 4a, the Horizontal Affinity Fields point towards the center of the lane line for each row of the output image. This is based on the valid lane pixels found in the binary segmentation output and represents the locations of potential lane instances with respect to all detected lane pixels. In fact, lane clusters are still successfully separated despite their respective Horizontal Affinity Fields being adjacent for numerous rows, demonstrated in yellow box of Figure 4a. Likewise, in Figure 4b, the Vertical Affinity Fields point along the lane towards the mean location of the next row’s lane pixels, which is based on the Horizontal Affinity Field outputs. This is visualized in the yellow box of Figure 4b, where for each unique lane instance, the unit vector points towards the next row’s mean lane pixel location. For both Figures 4a and 4b, the blue boxes clearly display how the Vertical Affinity Fields and Horizontal Affinity Fields are implemented on a single detected lane instance.

V. CONCLUDING REMARKS

In this paper, an approach to lane detection via lane instance segmentation through the use of binary segmentation masks and affinity fields was presented. The developed Horizontal and Vertical Affinity Fields along with the binary masks were demonstrated to successfully cluster lane pixels into unique lane instances as a post-processing step. This technique was implemented using a row-by-row decoding process with little overhead and is what enables LaneAF to detect a variable number of lanes without assuming a fixed/maximum number of lanes. Finally, this form of clustering is more interpretable in comparison to previous visual approaches and can be analyzed to easily identify and correct sources of error. Coupling this method with Deep Layer Aggregation, LaneAF achieves performance on par with the state-of-the-art on the TuSimple dataset.

TABLE II: Performance Comparison on TuSimple Test Set

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
<th>FP</th>
<th>FN</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-18</td>
<td>92.69%</td>
<td>0.0948</td>
<td>0.0822</td>
</tr>
<tr>
<td>ResNet-34</td>
<td>92.84%</td>
<td>0.0918</td>
<td>0.0796</td>
</tr>
<tr>
<td>ENet</td>
<td>93.02%</td>
<td>0.0886</td>
<td>0.0734</td>
</tr>
<tr>
<td>LaneNet</td>
<td>96.38%</td>
<td>0.0780</td>
<td>0.0244</td>
</tr>
<tr>
<td>SCNN [7]</td>
<td>96.53%</td>
<td>0.0617</td>
<td>0.0180</td>
</tr>
<tr>
<td>ENet-SAD [11]</td>
<td>96.64%</td>
<td>0.0602</td>
<td>0.0205</td>
</tr>
<tr>
<td>LaneAF</td>
<td>95.43%</td>
<td><strong>0.0235</strong></td>
<td>0.0460</td>
</tr>
</tbody>
</table>

In this paper, an approach to lane detection via lane instance segmentation through the use of binary segmentation masks and affinity fields was presented. The developed Horizontal and Vertical Affinity Fields along with the binary masks were demonstrated to successfully cluster lane pixels into unique lane instances as a post-processing step. This technique was implemented using a row-by-row decoding process with little overhead and is what enables LaneAF to detect a variable number of lanes without assuming a fixed/maximum number of lanes. Finally, this form of clustering is more interpretable in comparison to previous visual approaches and can be analyzed to easily identify and correct sources of error. Coupling this method with Deep Layer Aggregation, LaneAF achieves performance on par with the state-of-the-art on the TuSimple dataset.

REFERENCES

Fig. 5: Selection of results produced by LaneAF. Clockwise from top-left: multiple curved lanes with successful detection of an on-ramp on the right side; simple two-lane road; highway with identification of splitting lanes due to off-ramp; three-lane highway with merging lanes (also notable is the detection of a lane line due to a contrail).