Lane Extraction and Quality Evaluation: A Hough Transform Based Approach

Xingyao Wang[1], Da Yan[2], Ke Chen[3], Yancong Deng[4], Cheng Long[5], Kunlin Zhang[6], Sibo Yan[7]
[1]*University of Michigan,* xingyaow@umich.edu
[2]*The University of Alabama at Birmingham,* yanda@uab.edu
[3]*Huazhong University of Science and Technology,* chenkehust@hust.edu.cn
[4]*University of California, San Diego* yad002@eng.ucsd.edu
[5]*Nanyang Technological University,* c.long@ntu.edu.sg
[6]*Wuhan University,* 2017301200097@whu.edu.cn
[7]*KPMG LLP,* eric3ysb@gmail.com

***Abstract*—Lane recognition helps to guide vehicles and is important in driving assistance and autonomous driving. While deep learning techniques have advanced the accuracy of lane recognition, they only aim to minimize the pixel-wise difference between prediction and ground truth, which does not always reflect the true quality of lane recognition. The predicted road marking (white/yellow lines) pixels also do not directly reflect the lane parameters which are essentially straight lines. In this paper, a novel Hough transform based approach is proposed to simultaneously extract lane parameters and give an intuitive quality score on how the lane parameters match the ground truth. This approach can be used with any pixel-wise lane recognition algorithms.**

***Keywords*-Hough transform; lane detection; deep learning;**

I. Introduction

Traffic lane recognition now plays an important role in advanced driver-assistance systems (ADAS) which help drivers in lane keeping or give lane departure warning alerts. Lane parameters are also essential inputs to autonomous driving, a technology being actively developed nowadays.

Following the success of deep learning (esp. convolutional neural network or CNN in short) in achieving high accuracy in image classification, a number of efforts have been made to apply CNNs for semantic segmentation, which assigns a class label to each pixel in an image [8], [9], [2], [12], [13], [11], [10]. These models can be applied to camera-based lane recognition, where each pixel is assigned to either of the two classes: Lane or Background (hereafter, we simply **use "lane" to mean the road markings** like the solid/broken white/yellow lines that define lane boundaries, which are the actual line targets to extract), and the performance is evaluated as the pixel-wise difference between prediction and ground truth.

However, pixel-wise segmentation cannot produce information directly understandable by computers for decision making, such as lane parameters like the position and direction of a lane. We propose to use Hough transform [5] to extract lane parameters in polar coordinates from the prediction that output by a deep learning algorithm.

In addition, we notice that the pixel-wise difference between prediction output and ground truth may not always reflect the accuracy of the extracted lane parameters. For example, the predicted lane pixels may not match perfectly with ground truth lane positions due to (1) noises in images such as lanes occluded by vehicles or lanes not even painted on the road surface, and due to (2) slight shift of human-labeled lines from the actual lane positions, even though the extracted line parameters are pretty accurate.

Existing deep learning models tend to overfit towards noises in images and lane labels, since a pixel-wise loss function is used to evaluate the validation error during training, leading to superfluous iterations of gradient descent being executed after good model parameters have been reached. We thus propose a new Hough transform based metric that directly compares the difference between the parameters of the extracted lanes and those of the ground truth. There are 2 challenges in the metric design:

- Both prediction output and ground truth are pixels rather than lines. Our solution is to apply Hough transform to both of them to extract their line parameters for subsequent comparisons.
- Since an image may contain multiple lanes, the comparison should match each predicted lane with the corresponding one in the ground truth, which essentially is an instance segmentation problem. However, popular instance segmentation methods such as region proposals are preprocessing methods before inputting into a CNN model, while our evaluator requires a postprocessing approach for instance matching.

To tackle the second challenge, we propose to train a lightweight k-nearest neighbor (k-NN) classifier on the extracted lines from ground-truth annotation, which is used to match each predicted line with a specific lane. To obtain the lanes of the ground truth, we use k-means clustering to group the extracted lines, which are then input to the k-NN classifier for training where a special treatment is necessary to account for the imbalance of lane classes (see Section IV).

The rest of this paper is organized as follows. Section II reviews the related work on image segmentation with deep learning. Section III reviews how we extract lines from prediction output and ground truth. Section IV explains how we group the extracted lines by lanes, and how we match lane parameters from prediction output and ground truth to evaluate lane recognition quality. Finally, Section V reports our experimental results and Section VI concludes this paper.

II. Related Work

Lane detection systems have been studied for over two decades [7], but to reconcile a lane fitting model with an input image, existing algorithms rely on hand-crafted visual features [6]. Recent advancement in deep learning has generated a number of CNN models for image segmentation as pioneered by fully convolution network (FCN) [8], which can be used for lane detection without using hand-crafted visual features. FCN replaces FC layers at the end with 1×1 convolutional layers for dense prediction, which is then upsampled by deconvolution to generate the segmentation label map. However, since deep features lose spatial location information, existing work on CNN-based semantic segmentation focuses on techniques to remedy the loss.

FCN uses a skip architecture that fuses upsampled maps with output from shallower layers for further upsampling to avoid rough segmentation output. DeconvNet [9] further adds unpooling layers that remember the position of each maximum activation value when doing max pooling, which is used to place each activation back to its original pooled location. DeconvNet also uses region proposals from EdgeBox to help detect objects that are substantially larger or smaller than the receptive field. Other similar convolutional encoder-decoder architectures include SegNet [2] and U-Net [12]. DeepLab [4] uses atrous convolution (aka. dilated convolution) to enlarge the receptive field to incorporate larger context, and adopts atrous spatial pyramid pooling (ASPP) to account for different object scales. DeepLab also postprocesses the segmentation output by a fully connected conditional random field (CRF) to incorporate smoothness constraints in order to improve object delineation. CRF-RNN [13] avoids the need of postprocessing with CRF by modeling each iteration of mean-field approximate inference by common CNN operations, which are connected by an RNN structure for end-to-end learning.

In the context of lane detection, however, the above models may not be the best since a lane is a long object that spans road surface which requires a large receptive field to capture, and harsh conditions such as occlusion and raining make the detection even more difficult. Global convolutional network (GCN) [11] proposes to use large kernels that are able to cover entire objects, but instead of using an expensive dense $k \times k$ feature map, it uses a combination of $1 \times k + k \times 1$ and $k \times 1 + 1 \times k$ convolutions to reduce the number of model parameters, and also adds a boundary refinement (BR) module modeled as a residual structure to further refine the object boundaries. Spatial CNN (SCNN) [10] aims to capture spatial relationships of pixels across rows and columns of the same image by generalizing traditional deep layer-by-layer convolutions to slice-by-slice convolutions within feature maps where messages are passed as residuals, so that long continuous shaped structure or large objects can be captured even with weak appearance clues, such as lanes with occlusion and unpainted segments.

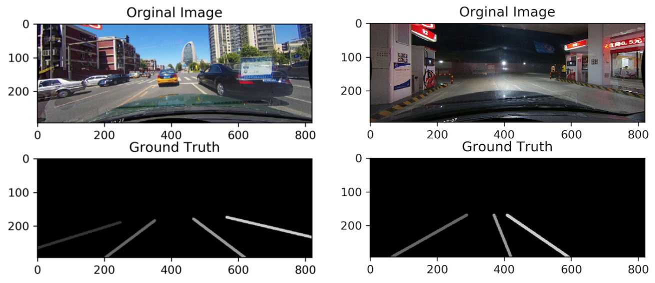

Figure 1. Samples from the CULane Dataset

Our Hough transform based evaluator can work with any deep learning model for semantic segmentation, though in this work we use GCN as a backbone network.

III. Line Extraction by Hough Transform

Recall that we want to extract a line for each lane in an image in terms of line parameter using Hough transform, so that the line can be used by subsequent decision making.

In a typical lane detection setting, video frames recorded by vehicle-mounted cameras are labeled by human annotators who can easily infer the lane positions and fill in the occluded part from the context, i.e., the viewable part of a lane. In this paper, we use the CULane [1] dataset which is created for academic research regarding traffic lane detection by the multimedia laboratory of the Chinese University of Hong Kong. The dataset includes video frames recorded by vehicles driven by different drivers in Beijing on different days. It contains 133,235 image frames including 88,800 in the training set, 9675 in the validation set, and 34,680 in the test set. See Figure 1 for two sample frames.

By training a GCN model over CULane, we obtain a segmentation model which takes an image and produces a prediction map of the same size as the input image. Each pixel x_i in the prediction map is associated with a value, i.e., the probability that x_i is on a lane, which equals $Pr(Lane) = sigmoid(s_{lane})$ where s_{lane} equals the lane-score for that pixel output by GCN (with a binary cross entropy loss).

Before extracting lines from the prediction map output by GCN using Hough transform, we first convert the prediction map into a binary image by thresholding: x_i is considered on a lane if $Pr(Lane) \geq 0.5$; otherwise, x_i is considered as in the background.

The binary image is then input to Hough transform to extract lines. Specifically, let (x, y) be a pixel on a lane as shown in Figure 2, then we can show that the blue line (for the lane) that passes through (x, y) can be represented as

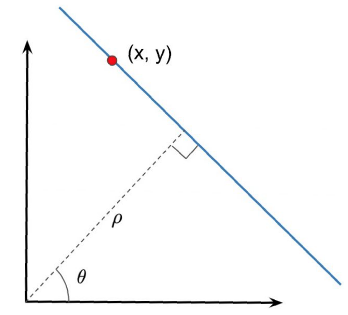

Figure 2. A Line in Polar Coordinates (ρ, θ)

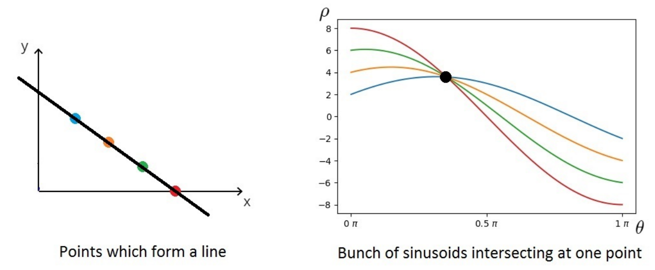

Figure 3. Line Detection by Hough Transform

$\rho = x \cos\theta + y \sin\theta$. Here, ρ is the distance from the origin to the closest point on the line, and θ is the angle between the x-axis and the line which connects with the origin and the closest point on the line. Different values of the Polar coordinates (ρ, θ) specify different lines that pass through (x, y), i.e., the lines rotate with center (x, y). In the (ρ, θ) system, these lines correspond to a sinusoid.

Now consider Figure 3 where four points on a lane are plotted on the left; the plot on the right shows the line functions that pass through these points (i.e., sinusoids), and note that they intersect at a particular (ρ, θ) point which represents the line of the lane that the four lane points are on.

To detect the lanes in a binary image, we can plot the sinusoid for each lane point in the (ρ, θ) coordinate system, and find the intersection points. Here, we use the “HoughLines” function of OpenCV which discretizes the (ρ, θ) coordinate system into cells and counts how many times each cell (ρ, θ) (which represents a line) is passed by a sinusoid. A threshold τ is used to find only those cells with counts at least τ, which correspond to the detected lines. For our dataset, we find that “HoughLines” works well in general when we use a discretization granularity of 1 pixel for ρ and 1 degree for θ, and use $\tau = 50$ to filter unlikely cells.

Now that we extracted lines that are likely to be on lanes in the form of a set $\{(\rho_1, \theta_1), (\rho_2, \theta_2), \ldots\}$, we need to group them by different lanes and to extract the line of each lane from its group. As we shall see in Section IV,

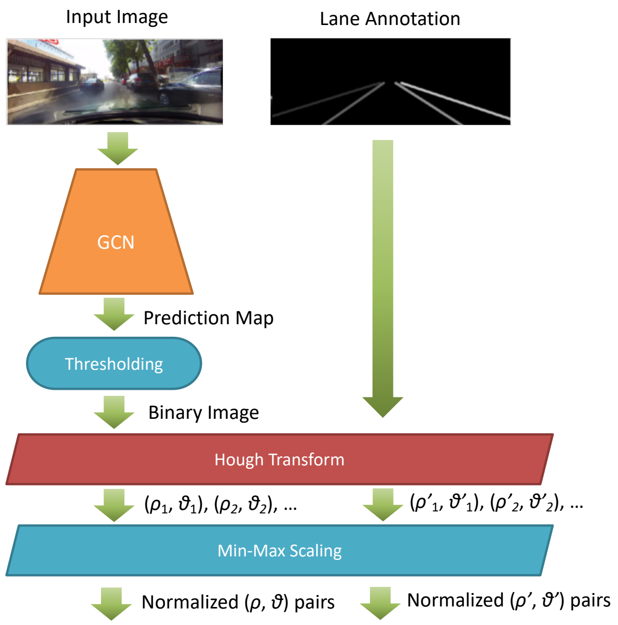

Figure 4. Line Extraction by Hough Transform

we perform this grouping using k-means clustering over $\{(\rho_1, \theta_1), (\rho_2, \theta_2), \ldots\}$, which works well since points of different lanes are properly separated. Since k-means (as well as the k-NN lane classification to be introduced in Section IV) works well when the data scale is normalized, we conduct min-max scaling of ρ and θ over every set $\{(\rho_1, \theta_1), (\rho_2, \theta_2), \ldots\}$ output by “HoughLines”. Our empirical study shows that the min-max scaling is important for the effectiveness of our approach.

Recall from Figure 1 (the two images at the bottom) that the ground truth annotations are already binary images. Therefore, unlike the GCN prediction map that requires binarization, we can apply “HoughLines” to directly extract lines $\{(\rho'_1, \theta'_1), (\rho'_2, \theta'_2), \ldots\}$ from a ground-truth annotation. As we shall see in Section IV, for each image, we extract lines from both the ground truth and the GCN prediction output, and then compares the two line sets to evaluate the lane recognition quality. As a nutshell, Figure 4 summarizes the operations described in this section.

IV. Lane Extraction and Matching

Recall from Figure 4 that we have extracted lines from the lane points of both the segmentation output and the ground-truth annotation. The next step is to (1) group the lines by lanes and extract a unique line from each group, and to (2) compare the lane-lines extracted from the prediction with those from the ground-truth annotation to evaluate the accuracy of lane recognition.

Since lines belonging to different lanes are usually well separated, it suffices to simply run the k-means algorithm on the normalized line parameters $\{(\rho_1, \theta_1), (\rho_2, \theta_2), \ldots\}$. We then take the mean or median (ρ, θ) of each group as the line for the corresponding lane. One issue is how to obtain the lane number k which is the clustering input:

- If we are evaluating the quality of the extracted lanes, e.g., during validation, we have access to the ground-truth annotation which has the number of lanes.
- If we are using our lane extraction approach during actual driving, then k can be obtained from GPS devices and online map services (e.g., Google Maps) which map our location to a road segment whose metadata usually contains the lane number. If such a data source is not available, we can use the popular elbow method to choose k which usually works well as the lane-lines are well separated.

One observation we obtain is that the binarized image of the GCN prediction map can sometimes be noisy due to lanes occluded by vehicles, lanes not painted on the road surface, and harsh weather conditions. In fact, in a lot of images, the lane positions are kind of "inferred" by GCN according to cars in nearby lanes thanks to the large kernel size of GCN. As a result, we recommend to use the median (ρ, θ) of each group from k-means clustering as the corresponding lane-line to minimize the impact of noisy lines. By taking the median, our Hough transform based line extraction approach has a denoising effect to obtain more robust lane-lines.

In contrast, if we are doing validation with ground-truth annotation at hand, we can obtain the lane-lines directly from the ground-truth annotation image that has minimal noise, in which case the mean (ρ, θ) of each group from k-means clustering serves the purpose well. Using these lane-lines as templates, we can then group the lines extracted from the GCN prediction map, e.g., by assigning each line to the group whose lane-line is the closest. This allows us to match the groups from the prediction with those from the ground truth according to their corresponding lanes for subsequent quality evaluation.

Instead of directly finding the closest lane-line obtained from ground truth, we match each line extracted from prediction to its specific lane by using a more robust k-NN lane classifier trained over the line groups returned by k-means over the lines extracted from the ground truth. In other words, we consider all lines extracted from the ground-truth rather than the group means (which are summary of those lines).

Figure 5 summarizes this process for each image in our validation set, where we can see that the extracted lines from ground truth are assigned group numbers using k-means clustering, which are then used to train a k-NN classifier. In contrast, the extracted lines from GCN output are directly input to the k-NN classifier to get their group numbers assigned. Note that the two line sets are matched with the same list of group numbers (i.e., class IDs) as given by k-means clustering over the lines from ground truth, and therefore groups are properly aligned by lanes for subsequent

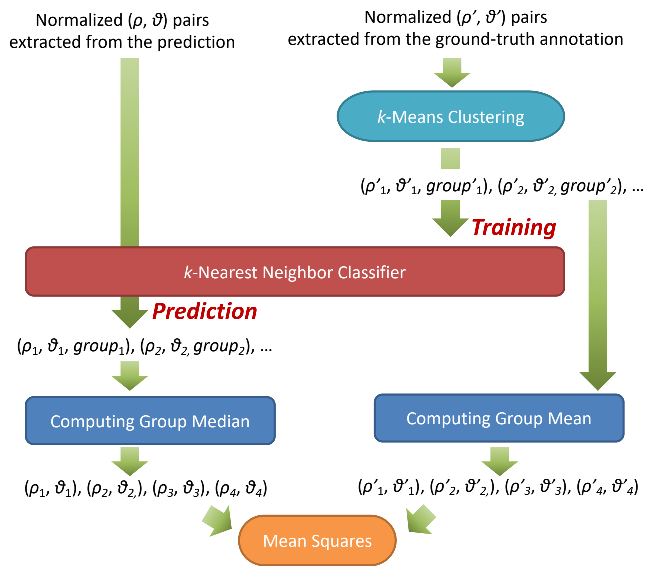
Normalized (ρ, ϑ) pairs extracted from the prediction
Normalized (ρ′, ϑ′) pairs extracted from the ground-truth annotation
k-Means Clustering
(ρ′1, ϑ′1, group′1), (ρ′2, ϑ′2, group′2), …
Training
k-Nearest Neighbor Classifier
Prediction
(ρ1, ϑ1, group1), (ρ2, ϑ2, group2), …
Computing Group Median
Computing Group Mean
(ρ1, ϑ1), (ρ2, ϑ2), (ρ3, ϑ3), (ρ4, ϑ4)
(ρ′1, ϑ′1), (ρ′2, ϑ′2), (ρ′3, ϑ′3), (ρ′4, ϑ′4)
Mean Squares

Figure 5. The Workflow of Our Evaluator

quality evaluation. Finally, we get the lane-lines for either line-set by computing the mean or median line parameters of each group (there are 4 lanes in Figure 5), and then compute the mean square error (MSE) of the two lane parameter vectors, which reflects how the lane-lines derived from GCN output are different from those extracted from ground truth. **This error is our evaluator of how far the detected lanes are different from the ground truth, which we call the HTB (Hough transform based) error.**

We omitted one small step for addressing the class imbalance problem in Figure 5 for simplicity, which is, however, important in making our approach work in practice. Let us assume that there are 4 lanes in an image. Since human annotations are of a good quality, points on a lane might hit on exactly the same cell in the (ρ, θ) coordinate system, leading to only one line detected by Hough transform. If we directly input the line set to a k-NN classifier, lines of this lane will be misclassified as the penalty in the training set is small. Figure 6 illustrates such a scenario where the red line-group from the ground truth only contains one line (Figure 6(a)), and using a k-NN classifier trained over such data, when classifying the lines extracted from GCN output, the lines of that lane are misclassified into the orange group (Figure 6(b)). Note that there are many lines of the red group extracted from GCN output, since the image data is noisy with, e.g., lane occlusions.

To address this problem, we adopt random over-sampling which randomly samples with replacement the current available lines in each under-represented group. We tried more advanced methods to over-sample minority classes such as SMOTE [3] but the performance is not better since our minority lane group often has only one line, and SMOTE generates synthetic lines by mixing the line with a nearby line which is often from another lane.

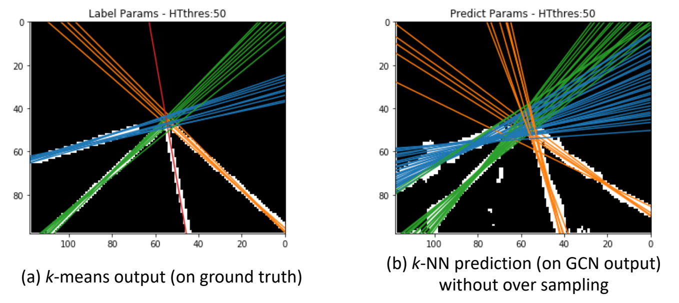

Figure 6. Misclassification of Minority Class

V. Experiments

We tested our approach on the CULane dataset [1] with GCN [11] as the backbone network for semantic segmentation. CULane contains 88,800 images for training, 9675 images for validation, and 34,680 images for test. With careful model selection, we find that setting the hyperparameter k of k-NN lane classifier as 3 gives the best performance and is robust. Our code is open-sourced at https://github.com/Xingyao-Wang/HT-Based-Evaluation-Metric.

We trained GCN over CULane while observing both our HTB error and the regular cross-entropy loss for pixel-wise segmentation, and saved the model parameters for subsequent experiments when the HTB error drops to a stable level. In fact, we find that if we run for more epochs, HTB error begins to increase slowly (indicating that the model begins to overfit) even though the regular cross-entropy loss is stable and keeps decreasing slowly.

A lot of pixel-wise evaluators have been used in the literature of semantic segmentation, such as mean square error (MSE), mean IoU (intersection over union), and F measure. We compared them with our HTB error and found that HTB error best reflects the quality of lane extraction. In the sequel, we report a comparison of HTB error with MSE (directly on pixels) as the representative pixel-wise evaluator.

Figure 7 shows a comparison between HTB error and direct MSE over our images in the validation set. The horizontal axis and vertical axis corresponds to HTB error and direct MSE, respectively, and the red line is where HTB error = direct MSE. We can see that most points are above the red line, which indicates that these images have a higher direct MSE than their HTB error. In other words, when the HTB error approaches zero which indicates a high accuracy in lane detection, direct MSE still has a high variance spread along the vertical axis. This is made clearer by Figure 8.

Also, in Figure 7, given the same direct MSE, say 0.1, HTB error can have a big variance which means that the lane parameter prediction accuracy can be either high or low on the same direct MSE level, meaning that pixel-wise MSE is not an accurate quality evaluator.

Finally, let us visualize the effectiveness of our Hough transform based (HTB) lane extraction. Figure 9 shows a three-lane scenario: the first image is the input image, and

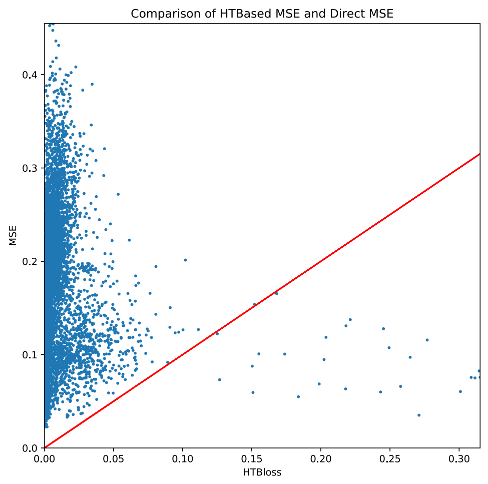

Figure 7. HTB Error v.s. Direct MSE

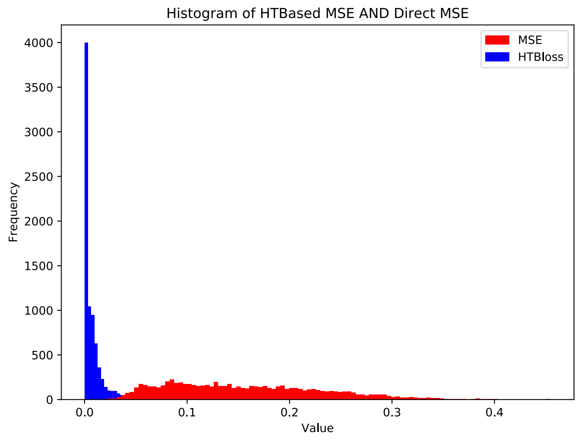

Figure 8. Error Distributions on Validation Set

white lines in the remaining 3 images are ground-truth lane pixels. In the second (resp. third) image, the red pixels reflect $Pr(Lane)$ output by GCN (resp. the lane pixels after binarization), where we can see that the white and red pixels are not perfectly overlapped. In fact, some detected lane pixels at the lower-right corner are noisy. The red lines in the fourth image are those output by our HTB extractor, which are very close to the white lines that are ground truth.

Figure 10 shows a two-lane scenario, where we can see similar observations: our HTB extractor effectively denoises the GCN output.

VI. Conclusions

This work proposed a novel Hough transform based lane extractor and the corresponding evaluator called HTB error which more accurately reflects the lane detection quality.

Acknowledgement. This work is partially supported by NSF DGE-1723250 and NSF OAC-1755464.

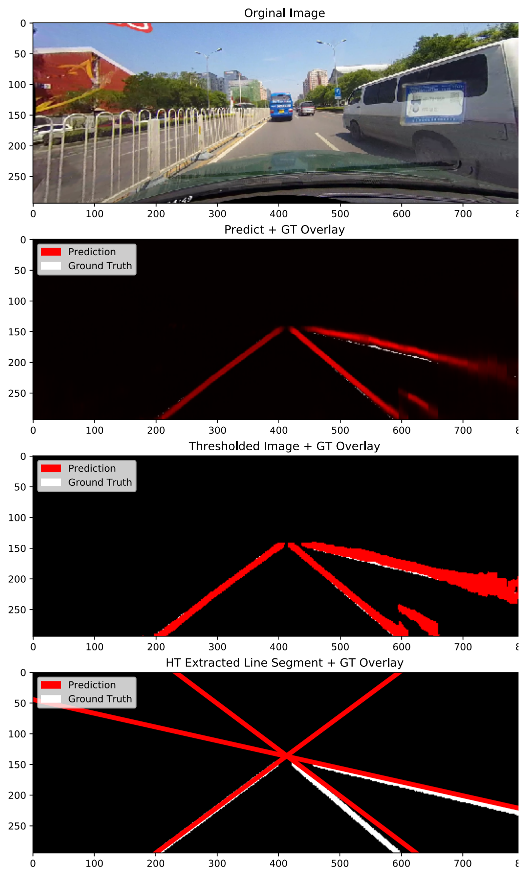

Figure 9. A Three-Lane Scenario

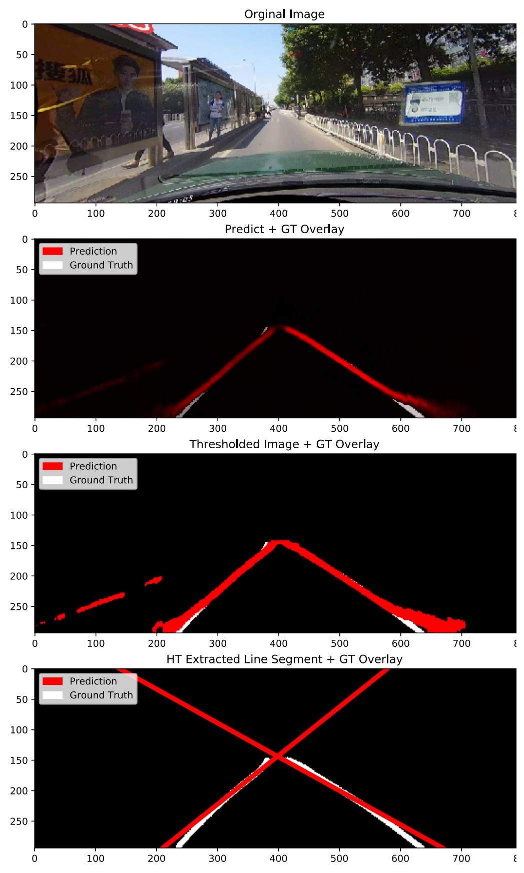

Figure 10. A Two-Lane Scenario

References

[1] CULane. https://xingangpan.github.io/projects/CULane.html.

[2] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(12):2481–2495, 2017.

[3] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *J. Artif. Intell. Res.*, 16:321–357, 2002.

[4] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(4):834–848, 2018.

[5] R. O. Duda and P. E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, 1972.

[6] A. Gurghian, T. Koduri, S. V. Bailur, K. J. Carey, and V. N. Murali. Deeplanes: End-to-end lane position estimation using deep neural networksa. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 38–45, 2016.

[7] K. Kluge and S. Lakshmanan. A deformable-template approach to lane detection. In *Proceedings of the Intelligent Vehicles' 95. Symposium*, pages 54–59. IEEE, 1995.

[8] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015.

[9] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *ICCV*, pages 1520–1528, 2015.

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

[11] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun. Large kernel matters - improve semantic segmentation by global convolutional network. In *CVPR*, pages 1743–1751, 2017.

[12] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241, 2015.

[13] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr. Conditional random fields as recurrent neural networks. In *ICCV*, pages 1529–1537, 2015.