[pages=1-last]filename.pdf

Robust Lane Detection Using Multiple Features

Tejus Gupta*, Harshit S. Sikchi* and Debashish Charkravarty

***Abstract*— Lane marker detection is a crucial challenge in developing self-driving cars. Despite significant research, large gaps remain between research and needs for fully autonomous driving. We highlight the limitations of present work and present a unified approach for robust and real-time lane marker detection. We present a multi-feature lane detection algorithm and give evidence why relying on one type of features can be harmful. We design a lane model using geometric constraints on lane shape and fit the lane model to the visual cues extracted. We improve the robustness of our algorithm by tracking lane markers temporally. We test our algorithm on KITTI dataset and show results that our algorithm can detect lane markers in presence of occlusions, sharp curves, and shadows.**

I. Introduction

Autonomous vehicles can significantly reduce the increasing number of traffic accidents caused by drivers fault [1]. In order to drive autonomously, the vehicle needs to segment drivable regions and lane boundaries as well as process this information for planning and control of the vehicle. This paper studies ego-lane detection where the task is to identify the lane the vehicle is currently driving on. There are several challenges associated with ego-lane detection the most critical of which are as follows [2]

1. Diversity in road and lane marker appearance (e.g, curved and dashed lanes)
2. Occlusions due to near-by vehicles
3. Intensity variations on road (shadows from nearby trees, buildings and, other vehicles that create misleading edges and texture on the road, glare, etc).

Several approaches have been followed for lane detection[3], [2]. However, none of the methods have successfully overcome all the challenges described above. Most lane detection methods extract specialized features and use these features to fit a lane model.

[4], [3] and [5] extract features based on vanishing point constraint. Due to perspective projection, road boundaries in the image plane must meet on a shared vanishing point on the horizon. Wang et al [4] use CHEVP (Canny-Hough Estimation of Vanishing Points) algorithm for initializing their B-Snake lane model. CHEVP detects edges using Canny edge detection and uses hough lines to fit lines. They compute the intersection of every pair of lines to vote for vanishing point estimate. Then, they assume that the lines voting for the detected vanishing point are road boundaries and use these lines to initialize their lane model parameters. Yoo [3] focused on improving vanishing point detection by using probabilistic voting in his work. Kang [5] used the position of vanishing point to generate hypothesis for the position of adjacent lanes. Vanishing point based methods have a disadvantage that line detection suffers from high false detection rate under variable scene illumination. Parajuli [6] describes this problem and uses local gradient spectrum to detect lane in shadows and low-illumination conditions. Other works have used color-based features, ridge features, etc.

There have been various approaches for modeling lane boundaries. Wang [4] uses uniform cubic Bezier-splines for

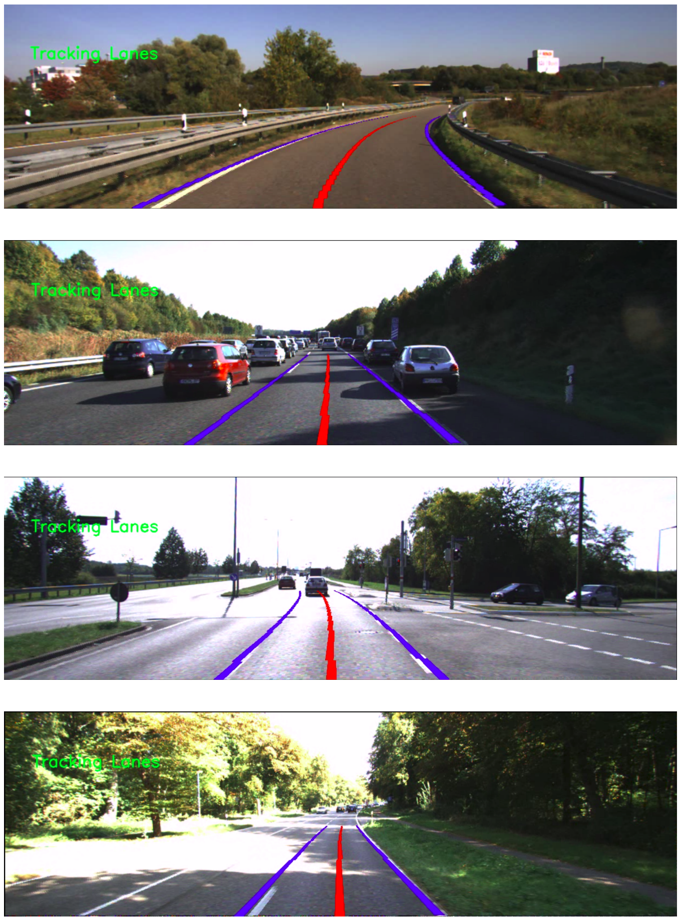

Fig. 1: Results on the Kitti-Dataset. Blue curves are lane boundaries and the red curve is the lane center line.

* Denotes equal contribution

T. Gupta, H. Sikchi and D. Chakravarty are with the Autonomous Ground Vehicle Team, Indian Institute of Technology Kharagpur, WB, 721302, India tejusgupta14@gmail.com, hsikchi@cs.cmu.edu, dc@iitkgp.ac.in

modeling lane markers. They initialize their model using CHEVP algorithm and track the control points for Bezier-spline temporally using an energy minimization framework.

Aly [7] also uses splines to model the lane marker and uses RANSAC for spline fitting. McCall et al. [8] used a simple parabolic model which incorporates lane position, angle, and curvature. Most algorithms [4], [6], [3] assume that lane markers are straight and so their lane model is incapable of detecting curved lanes. There is a trade-off between over-constrained models which do not cover all existing road geometries and under-constrained ones which tend to over-fit noisy features. Lane detection results have been improved by integrating knowledge from previous frames using Kalman filter [9], [10], extended Kalman filter [11], [12], particle filter [13], [14], and super-particle filter [15].

In general, these approaches are prone to robustness issues and recent methods have used deep neural networks to detect lane markers. Convolutional neural networks designed for semantic segmentation [16] can be trained to obtain pixel-wise lane segmentation. [17] posed lane detection as an instance segmentation problem with each lane as an instance and proposed a network architecture that can be trained end-to-end for this task. In [18], a spatial CNN architecture was suggested that passes information slice-by-slice within each feature map and is able to exploit the strong shape prior of lane markings. [19] improved performance by jointly training for estimating vanishing point and detecting objects with shared weights.

We treat semantic segmentation from deep neural networks as specialized feature and use the lane model and temporal tracking to improve robustness.

In this paper, we present a robust approach to ego-lane detection using monocular RGB images. The contributions of this paper can be summarized as follows:

1) An ego-lane detection algorithm is proposed that performs robustly in case of curved roads, occlusion, and shadows. The design is aimed to use several low-level visual features so that the detector doesn't fail when one of the visual features gives incorrect results. Potentially a significant number of features extracted may be incorrect or noisy and so a robust estimator is used for updating the lane model at every step.

2) The proposed method's performance is evaluated on the KITTI dataset. A novel evaluation metric is proposed which is more suitable for judging lane detection results.

II. Proposed Methodology

The first step in our lane detection pipeline is lane features extraction. In this context, a feature is a region of the image that is a suspected lane marker position. Since this step has a high false positive rate, we fit a geometric model to the visual features extracted. Our fitting procedure rejects outliers and gives a compact high-level representation of the lane markers. Finally, we fuse our detection result with knowledge from previous frames to improve detection accuracy.

A. *Lane Features Extraction*

We use multiple algorithms for extracting lane features. Each of these algorithms exploits different properties of lane markers. This ensures that our lane detection algorithm doesn't fail when one of the feature extractors gives incorrect results. For example, edge-based feature detectors don't work well in shadows and so lane detection algorithms that solely rely on edge-based features fail in low-illumination conditions.

1) ***Gradient-based features:*** We exploit the property that lane markers form intensity edges. Line segments are extracted using line segment detector (LSD) method in [20]. We use LSD method because it reduces false detections which often arise in Hough transform. LSD performs well without parameter tuning. After this step, the detected line segments contain several non-lane features and so geometric constraints are used to remove these.

All parallel lines in a plane meet at a fixed line in image plane when viewed from a camera using pin-hole perspective model. Assuming a flat road, this property implies that lines corresponding to lane markers intersect on a fixed line in the image plane i.e. horizon. We now describe an algorithm that uses this property to remove non-lane lines detected.

For each line segment $\mathrm{L_i}$, the intersection point with every other line segment is considered and used to compute a score for $\mathrm{L_i}$.

$$score(L_i) = \sum_{j \subset S(L_i)} length(L_j)$$

$$S(L_i) = \{\, j \mid \mathrm{L_i} \text{ and } \mathrm{L_j} \text{ intersect at horizon}\}$$

All lines with a score below the threshold are removed. After this step, only lines corresponding to lane markers and noisy lines that are parallel to road boundary are retained.

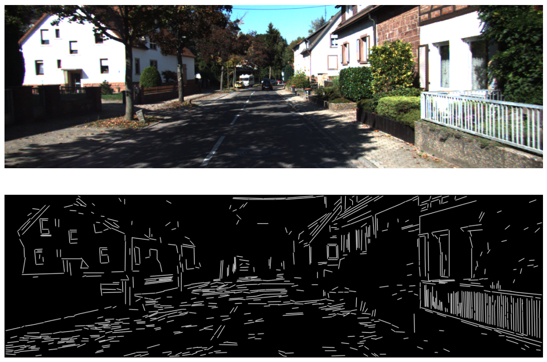
Fig. 2: Image with heavy shadows and edges detected by LSD.

2) ***Intensity-based features:*** Shadows create misleading edges and texture on the road and hence edge features fail to work robustly in presence of shadows. Fig. 2 shows how shadows mask lane-road edges and cause false intensity edges. In different illumination conditions (e.g, in presence

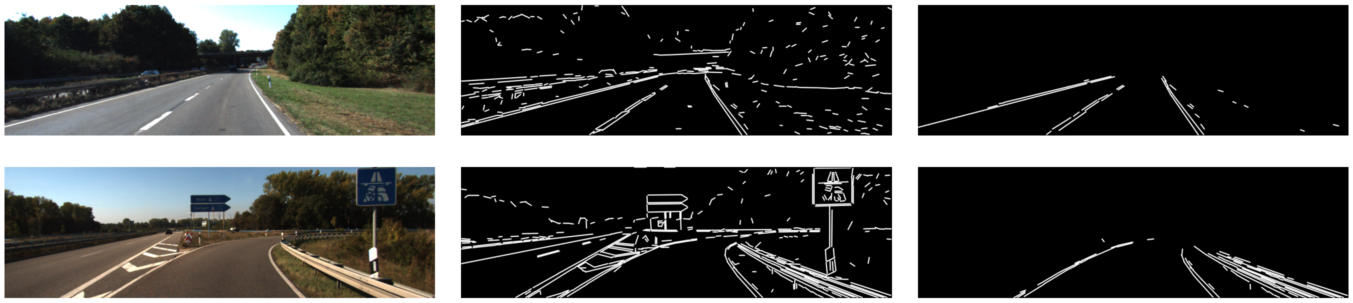

Fig. 3: From left to right: Input image, edges detected using LSD, filtered lines obtained using the horizon constraint.

of shadows), lane markers may have different brightness yet maintain their superiority relationship with their horizontal neighbors. So, lane markers can be detected by searching for low-high-low pattern horizontally in the image.

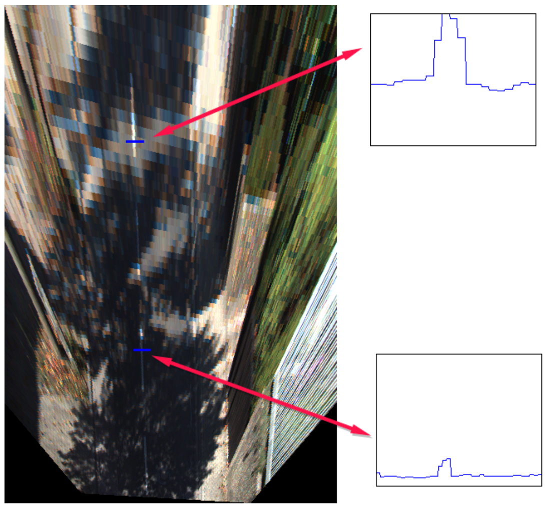

Fig. 4: Top-view image and intensity profile with/without shadow.

Due to perspective distortion, searching for this pattern is difficult as lane width varies with image row. We circumvent this problem by generating the top-view of the road image. In this inverse perspective image, the lane marker width is constant. To get the inverse perspective transform of the input image, we assume a flat road and compute the homography matrix using the camera intrinsic (focal length and optical center) and extrinsic (pitch, yaw, and height above ground).

The top-view image is smoothed vertically and convoluted with the second derivative of Gaussian. We tune standard deviation (σ) of the Gaussian to respond to bright vertical lines of specific width on a dark background. Since the orientation of lane marker isn't necessarily vertical, we apply this filter at five orientations ($-15°$, $-7.5°$, $0°$, $7.5°$, $15°$) and take pixel-wise maximum of the filter response.

The output of this step provides an accurate estimation of lane markers but we still need to threshold the filter output to remove noisy response. Our thresholding method is similar to the hysteresis thresholding technique used in Canny edge detector. We select low and high threshold values, retaining the pixels whose value lies above the high threshold and those below the low threshold are rejected. For pixel values lying between this range, a graph search is performed and the diameter of this graph is computed. Pixels for this graph are accepted if the diameter exceeds a minimum threshold. This minimum threshold corresponds to minimum lane marker length encountered in the bird's eye image. This threshold is set considering the dashed lanes faced in various scenarios.

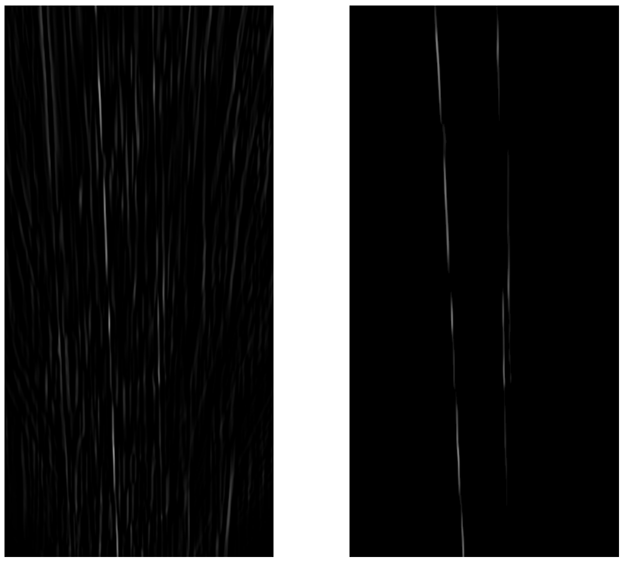

Fig. 5: (Left) Top-view image after convolution, (Right) Image after thresholding.

*3) **Texture-based features**:* Roads are not always bounded by man-made markings e.g, dirt roads have no marking at all and only color or texture difference between the road and off-road areas can indicate the boundaries. We identify road region and use the boundaries as our third feature. Semantic segmentation is well-researched problem and recent years have experienced rapid progress, largely due to new deep learning based methods.

In our pipeline, we use the Multinet model [21] for road segmentation. The MultiNet architecture uses an encoder-decoder architecture with shared encoder for classification, detection, and semantic segmentation. It achieves state-of-

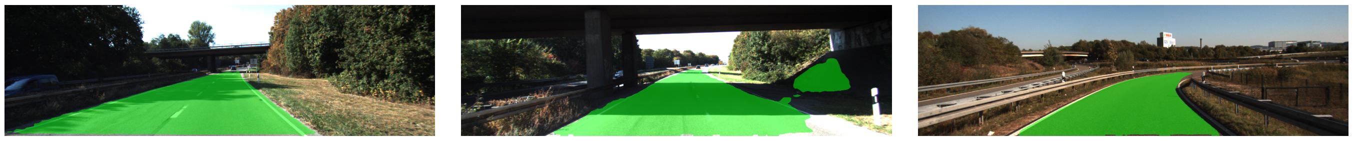

Fig. 6: Segmentation results using MultiNet on images from KITTI sequence.

the-art results with 92.2% mean precision in road segmentation on KITTI dataset and takes 43 ms for inference.

B. Use previous frame's lane estimate to filter extracted features

We reduce computation and prevent erroneous detection by limiting the image regions from which features are extracted. In our implementation, we only keep features that are less than 2.5 meters away from previous lane estimate. This step also helps us identify which features correspond to left and right lane markers.

We choose the threshold of 2.5 meters using our vehicle's speed constraints. Our car has a maximum speed of 60 km/hr and our algorithm runs at 10-15 frames per second, so the car moves at most 1.5 meters between consecutive frames.

C. Lane Model Estimation

Lane model plays an important role in lane detection. The model represents our assumption about the shape and geometry of lane markers in the real world. The lane model also helps us to extract a compact representation of the scene. This representation can be used for more complex road understanding e.g, identifying lane merges and splits and higher level decision making.

For our ego lane model, we assume the lane markers are parallel on the ground plane. We approximate the lane center line with a cubic polynomial.

$$\mathrm{P} = \mathrm{ax}^3 + bx^2 + cx + d$$
$$\mathrm{P}_L = ax^3 + bx^2 + cx + d - w/2$$
$$\mathrm{P}_R = ax^3 + bx^2 + cx + d + w/2$$

An advantage of our lane model is that it enables us to detect both lanes even if one lane is heavily occluded. We can fit our lane model with points from only one lane if the lane width is known. In our implementation, we do not update the lane width if number of points obtained on either of the lanes is below a threshold.

*1) **Initialization:*** The initialization method consists of two steps: a simple global estimation of lane position and refining our estimate using RANSAC. A histogram is created using the column-wise sum of feature points and smoothing it using Gaussian filter. The histogram is convolved with a filter described below.

$$T(w) = \begin{cases} 1 & x \le w/2 + k \ and \ x \ge w/2 - k \\ 1 & x \le -w/2 + k \ and \ x \ge -w/2 - k \\ 0 & otherwise \end{cases}$$

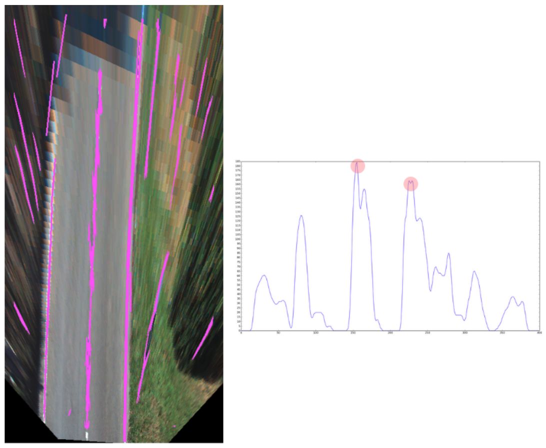

Fig. 7: Features extracted and histogram showing column-wise sum of extracted pixels. The red circles are the lane position estimate.

The filter is similar to the one used for extracting intensity features but responds to low-high-low patterns separated by distance w, which is the separation between lanes. This filter prefers the lane boundaries separated by width w with some error specified by 'k' in the above formula. We use w = 3.7 meter which is the lane width standard in Europe. The maxima of response with the above filter are used as lane center and lane positions are estimated at the positions of the local maxima.

We use a window of size 2 meter around the lane position estimate and features in this window are used to find the exact lane position and orientation. We use RANSAC spline fitting to estimate the parameters of the lane model. This estimate is used to initialize our tracking step.

*2) **Outlier Elimination and Model Update:*** Potentially there are a significant number of non-lane points among the visual features extracted. Because of the presence of these outliers, the standard method of least squares estimation is not suitable. The aim is then to obtain a set of inliers consistent with the lane geometry using a robust estimator.

The RANSAC algorithm selects the hypothesis with the highest number of inliers. It finds the minimum of the following cost function

$$C = \sum_i \rho(e_i^2)$$

where

$$\rho(e_i^2) = \begin{cases} 0 & e^2 < T^2 \\ constant & e^2 \geq T^2 \end{cases}$$

and e_i is the cost function for the ith data point, and T is the threshold for considering a data point as inlier.

In MSAC [22], Torr and Zisserman proposed to truncate the error instead of thresholding to gain additional sensitivity at no additional cost.

$$C^{'} = \sum_i \rho^{'}(e_i^2)$$

where

$$\rho^{'}(e_i^2) = \begin{cases} e^2 & e^2 < T^2 \\ T^2 & e^2 \geq T^2 \end{cases}$$

We use MSAC to identify the set of inliers. In every frame, the parameters of the lane model are updated with the least squares fit to the inliers.

Algorithm 1 Lane Detection using Multiple Features

```
begin
    Extract visual features.
    Initialize lane model.
    for every new frame do
        Extract visual features in search region defined
        using previous lane estimate.
        Identify inliers using MSAC.
        Update model with least squares fit for set of inliers.
    end
end
```

III. Experimental Results

A. Dataset

We chose the KITTI autonomous driving dataset [23] as the testbed for our evaluation since it provides a large variety of challenging, real-world images of different scenarios. The KITTI lane detection benchmark has images annotated with the ego-lane marking but is not suitable for evaluating our algorithm as the provided images aren't continuous and our algorithm tracks lane estimates temporally.

The 'Road' category[1] contains twelve different sequences obtained in the daytime and capturing a wide range of interesting scenarios. We have annotated four challenging sequences from the 'Road' category and used it to test our detector. The sequences are suitable for testing the detector in presence of heavy traffic, curved roads, shadows, and glare on roads. In all experiments, we limit ourselves to a thorough evaluation of the ego lane detection.

We manually annotated ego lane boundary up to 80 meters from the vehicle for each frame in these sequences. The annotated dataset contains 1850 images. We use a tool that fits a cubic spline to lane markers in Birds Eye View for annotation. We manually verified that the lane boundary marked were satisfactorily accurate.

B. Evaluation Metric

Previous methods [7], [3] have used image-based evaluation metrics like average localization errors in pixels. We choose to evaluate our algorithm in the Bird's Eye View space as vehicle controls need the lane boundaries in this frame. Unlike previous image-based methods, the results of our detector can be compared to lane detection performed using other sensors like LIDAR.

The detection results shown in Table II are computed automatically using the hand-labeled data. To calculate detection rate, in each frame, each detected lane boundary is compared to ground truth lanes, and a check is made to decide if it is a correct or false detection. For the detected curve to be correct, the maximal error from ground truth must be less than t.

$$\forall P_i \in S : max(dist(P_i, C_j)) < t$$

where, S is the set of points in the annotated groundd truth and *Cj* is the nearest point to *Pi* on the detected curve.

The mean error in centimeters is calculated as the average distance of points in the ground truth annotation with the corresponding closest point on the detected curve.

$$\sum_i dist(P_i, C_j)/T$$

where T is the total number of points belonging to lane in ground truth and *Cj* is the nearest point on the detected curve to the corresponding *Pi* on S.

Sequence Name [Kitti[1]]	Frames	Conditions
2011_09_26_drive_0027	384	Heavy Shadows
2011_09_26_drive_0032	396	Glare on Road
2011_10_03_drive_0042	400	Curved Road
2011_10_03_drive_0047	670	Heavy Traffic

C. Results

Table II shows the performance of our lane detection algorithm in diverse conditions encountered in KITTI Dataset. The detection rate is computed with t = 70 cm. The proposed algorithm works well in the case of shadows, curved and dashed lanes, heavy traffic, illumination conditions which were the challenges we had described before. The algorithm also runs in real time at 10 frames per second on an Intel core I7-5500 2.40 GHz processor and Pascal Titan X GPU. Test Results on KITTI Dataset are shown in the link [2].

IV. Conclusion

In the paper, we present a unified lane marker detection algorithm. Our algorithm is able to detect lanes in a wide range of scenarios robustly.

[1]http://www.cvlibs.net/datasets/kitti/raw_data.php?type=road

[2]https://youtu.be/TzlBsWBXbVs

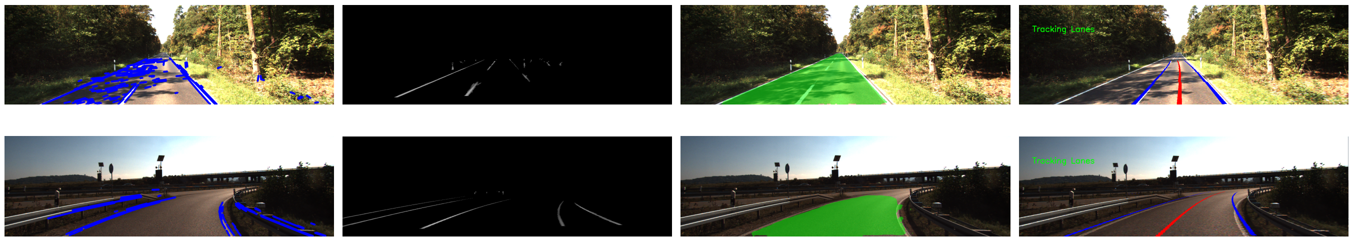

Fig. 8: From left to right - gradient-based features, intensity-based features, texture-based features, lane detection result. In both examples, some features are noisy/incorrect but the lane detector performs well.

TABLE I: Evaluation for ego-lane detection

Sequence Name [Kitti[1]]	Detection Rate	Mean Error (cm)
2011_09_26_drive_0027	83%	54.7
2011_09_26_drive_0032	91%	34.5
2011_10_03_drive_0042	94%	53.1
2011_10_03_drive_0047	87%	44.1

Our future work will study the usefulness of using other sensors for extracting lane features e.g, stereo camera and 3D lidar, as well as explicit lane marker segmentation using deep neural networks.

REFERENCES

[1] M. Ruikar *et al.*, "National statistics of road traffic accidents in india," *Journal of Orthopedics, Traumatology and Rehabilitation*, vol. 6, no. 1, p. 1, 2013.

[2] A. B. Hillel, R. Lerner, D. Levi, and G. Raz, "Recent progress in road and lane detection: a survey," *Machine vision and applications*, vol. 25, no. 3, pp. 727–745, 2014.

[3] J. H. Yoo, S.-W. Lee, S.-K. Park, and D. H. Kim, "A robust lane detection method based on vanishing point estimation using the relevance of line segments," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 12, pp. 3254–3266, 2017.

[4] Y. Wang, E. K. Teoh, and D. Shen, "Lane detection and tracking using b-snake," *Image and Vision computing*, vol. 22, no. 4, pp. 269–280, 2004.

[5] S.-N. Kang, S. Lee, J. Hur, and S.-W. Seo, "Multi-lane detection based on accurate geometric lane estimation in highway scenarios," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, 2014.

[6] A. Parajuli, M. Celenk, and H. B. Riley, "Robust lane detection in shadows and low illumination conditions using local gradient features," *Open Journal of Applied Sciences*, vol. 3, no. 01, p. 68, 2013.

[7] M. Aly, "Real time detection of lane markers in urban streets," in *Intelligent Vehicles Symposium, 2008 IEEE*, pp. 7–12, IEEE, 2008.

[8] J. C. McCall and M. M. Trivedi, "Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation," *IEEE transactions on intelligent transportation systems*, vol. 7, no. 1, pp. 20–37, 2006.

[9] A. Borkar, M. Hayes, and M. T. Smith, "Robust lane detection and tracking with ransac and kalman filter," in *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pp. 3261–3264, IEEE, 2009.

[10] . Obradović, Z. Konjović, E. Pap, and I. J. Rudas, "Linear fuzzy space based road lane model and detection," *Knowledge-Based Systems*, vol. 38, pp. 37–47, 2013.

[11] K. Zhao, M. Meuter, C. Nunn, D. Müller, S. Müller-Schneiders, and J. Pauli, "A novel multi-lane detection and tracking system," in *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pp. 1084–1089, IEEE, 2012.

[12] B. Özcan, P. Boyraz, and C. B. Yiğit, "A monoslam approach to lane departure warning system," in *Advanced Intelligent Mechatronics (AIM), 2014 IEEE/ASME International Conference on*, pp. 640–645, IEEE, 2014.

[13] Z. Kim, "Robust lane detection and tracking in challenging scenarios," *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, no. 1, pp. 16–26, 2008.

[14] H. Li and F. Nashashibi, "Robust real-time lane detection based on lane mark segment features and general a priori knowledge," in *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*, pp. 812–817, IEEE, 2011.

[15] B.-S. Shin, J. Tao, and R. Klette, "A superparticle filter for lane detection," *Pattern Recognition*, vol. 48, no. 11, pp. 3333–3345, 2015.

[16] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

[17] X. Pan, J. Shi, P. Luo, X. Wang, and X. Tang, "Spatial as deep: Spatial cnn for traffic scene understanding," *arXiv preprint arXiv:1712.06080*, 2017.

[18] D. Neven, B. De Brabandere, S. Georgoulis, M. Proesmans, and L. Van Gool, "Towards end-to-end lane detection: an instance segmentation approach," *arXiv preprint arXiv:1802.05591*, 2018.

[19] S. Lee, J. Kim, J. S. Yoon, S. Shin, O. Bailo, N. Kim, T.-H. Lee, H. S. Hong, S.-H. Han, and I. S. Kweon, "Vpgnet: Vanishing point guided network for lane and road marking detection and recognition," in *Computer Vision (ICCV), 2017 IEEE International Conference on*, pp. 1965–1973, IEEE, 2017.

[20] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "Lsd: A fast line segment detector with a false detection control," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 4, pp. 722–732, 2010.

[21] M. Teichmann, M. Weber, M. Zoellner, R. Cipolla, and R. Urtasun, "Multinet: Real-time joint semantic reasoning for autonomous driving," *arXiv preprint arXiv:1612.07695*, 2016.

[22] P. Torr and A. Zisserman, "Robust computation and parametrization of multiple view relations," in *Computer Vision, 1998. Sixth International Conference on*, pp. 727–732, IEEE, 1998.

[23] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.