Lane detection using lane boundary marker network with road geometry constraints

Hussam Ullah Khan, Afsheen Rafaqat Ali, Ali Hassan, Ahmed Ali, Wajahat Kazmi, Aamer Zaheer
KeepTruckin Inc. (RnD)
Lahore, Pakistan
{hussam.khan, afsheen, ali.hassan, ahmed.ali, wajahat.kazmi, az} @keeptruckin.com

Abstract

Lane detection is of critical importance to both the self-driving cars as well as advanced driver assistance systems. While current methods use a range of features from low-level to deep features extracted from convolutional neural networks, they all suffer from the problem of occlusion and struggle to detect lanes with low or no evidence on the road. In this paper, we use a lane boundary marker network to detect keypoints along the lane boundaries. An inverse perspective mapping is estimated using road geometry which is then applied to the detected markers and lines/curves are fitted jointly on the rectified points. Finally, missing lane boundaries are predicted using lane geometry constraints i.e., equidistant and parallelism. Reciprocal weighted averaging ensures lane boundaries with strong evidence dominate their predicted alternatives. The results show a significant improvement of +7.8%, +6.8% and +1.2% of F1 scores over the state-of-the-art on CULane, Caltech and TuSimple datasets, respectively. This proves our algorithm's robustness against both occluded and missing lanes cases. Furthermore, we also show that our algorithm can be combined with other lane detectors to improve their lane retrieval potential.

1. Introduction

Fully autonomous vehicles in the form of self-driving cars are gradually appearing on the scene and we have Google Waymo [17], Tesla's Auto-pilot [19], Comma.Ai's OpenPilot kit [16] and Intel's MobileEye solutions [18] just to name a few. Along with self-driving cars, Advanced Driver Assistance Systems (ADAS) have also captured a surge in the interest. Both self-driving cars and the ADAS require different levels of autonomy but some aspects of automation are still common. One of them is road lane detection and segmentation. Lane detection is critical in identifying and ensuring safe driving practices and an on-board system with this capability can alert the driver in case of an unsafe lane change and avoid it in the first place if it is a self-driving vehicle. Besides, it is also necessary in understanding the road layout and surrounding scene geometry for proper route planning for autonomous or semi-autonomous vehicles.

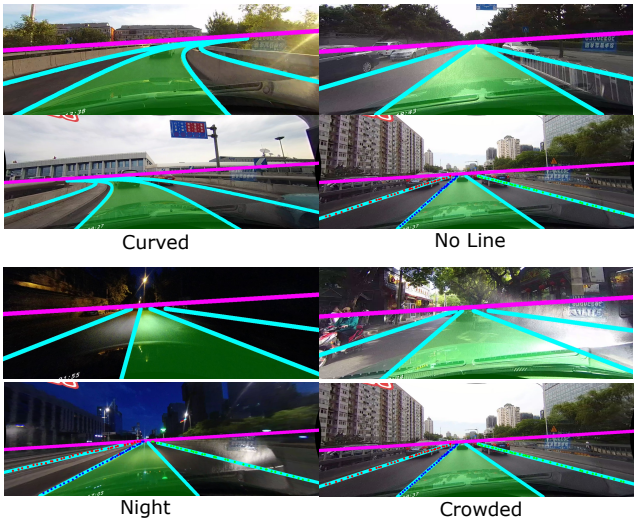


Figure 1. Sample results of our algorithm on examples from four different classes of CULane dataset [33] are shown here. Cyan lines are the detected lane boundaries, green region represents the ego lane and magenta line displays the estimated horizon. In the *No Line* class, there is actually no line markings on the road but the ground truth carries the lines shown.

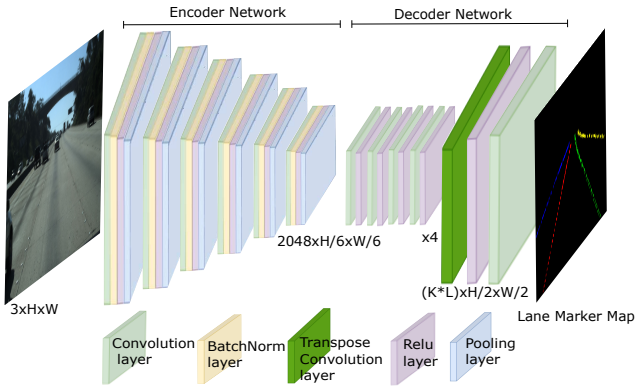
Lane detection has classically been done using either low-level edge features, histogram analysis, lane boundary detection algorithms such as Linear Hough Transform [15] or a combination. Although such methods are intuitive and efficient, they are however easily fooled by non-lane edges or lines appearing in the images such as road curbs, shadows as well as false negatives due to erased or inconsistent lane boundary markings. Recently, deep learning based solutions have also been proposed that produce more robust

and reliable results as they tend to learn the context of the road scene irrespective of occlusions. The advantage of using deep learning (DL) techniques lies in the discourse from traditional lane boundary detection approach towards semantic segmentation of the lane region [31]. Several state-of-the-art algorithms such as VPGNet [24], SCNN [33] and JLBNet [40] have shown impressive results and have set the bar very high. But there are challenging scenarios in which even DL based algorithms struggle to find the lane. One of the obvious cases is when the evidence of a lane line is either effaced or there is no lane marking on the road but the ground truth (GT) carries lane information such as CULane's [33] *no line class*; examples are shown in Figure 1.

To address such cases, we proposed a novel lane detection algorithm that exploits the parallel nature of lane boundaries and fixed lane widths on the road to validate the detected lanes and predict lanes that are missed by the detector due to erased or effaced lane markings. In general, these assumptions apply on all through lanes that typically constitute the major length of the highway and our main focus in this study was to evaluate the effectiveness of the proposed approach in such scenarios. Although, the assumption of parallel lanes has been successfully exploited in earlier works to detect lanes in rectified view [5] but to the best of our knowledge, we are the first ones who used equidistant and parallelism property of the lanes jointly to fit the lanes and our results on *no line class* of CUlane dataset highlight the effectiveness of this approach.

We use a simple encoder-decoder based Convolutional Neural Network (CNN) to detect keypoints or markers (used interchangeably) along the lane boundaries. Inverse Perspective Mapping (IPM) estimated using scene geometry is then applied to the detected markers and lines/curves are fitted on the rectified points. The estimation of IPM is a one-off calculation for any given camera which is done over a small set of initial frames. Unlike earlier approaches that used single lane evidence to fit a curve for each lane line we exploited road geometry and jointly fitted lines/curves for all lanes. Our approach significantly improved the results on public datasets, specifically for crowded or missing lane scenarios. Lane region segmentation would descend from lane boundary detection as a boundary polygon. The sample results of our lane detection algorithm on four different classes of CULane dataset, shown in Figure 1, demonstrate the efficacy of our approach.

The breakdown of the paper is as follows: Section 2 provides a quick review of currently explored lane detection techniques. Section 3 discusses our approach in detail with our keypoint detector in section 3.1 and the estimation of horizon and IPM in section 3.2.1 and 3.2.2. Our main contribution, the parallel lane fitting along with the missing lane prediction algorithm, is discussed in sections 3.3.1 and 3.3.2 respectively. Section 4 outlines the datasets, training and testing frame work. Results are presented in sections 4.1, 4.2 and 4.3. Section 5 concludes the paper.



(a)

(b)

Figure 2. (a) The figure shows the architecture of the lane boundary marker network. (b) The sampled keypoints from the ground truth lane line are shown here.

2. Related Work

In literature, the problem of lane detection has been addressed through a variety of sensor data. Taking into account the variation of ambient light owing to the time of the day and weather which may render passive color imaging useless, active sensors such as radars and lidars (Light/Laser Detection and Ranging) have also been explored [21, 9, 26]. With this approach, generally the white/colored lane boundary is identified in the range point cloud as having a different reflectance w.r.t the road surface. The identified lane boundary is then used to fit a line model on top of it. But this is not a very popular approach since the Lidar can be expensive with complex installation along with the challenges posed by road surface conditions such as the fading color of the lane boundary over time which is quite common across the globe.

The most straightforward approach, therefore, is using a forward looking monocular dashcam installed inside the driver cabin. Generally, images thus acquired are processed for low-level features. Veit *et al*. [37] presented a comprehensive overview of features used to detect road markings up till 2008. Most commonly, edge detection filters such as Canny and Sobel [32, 20, 38] were employed for detecting lane boundaries followed by model based approaches such as Hough Transform [6]. Bertozzi *et al*. used Inverse Perspective Mapping to rectify the image removing the per-

spective distortion to improve lane detection by employing parallel line constraint of the road lane boundaries [5]. But under all the constraints, these methods still suffered from occlusion and the false negatives generated by strong shadows. Vanishing Point (VP) constraint was imposed in [29, 6] to reduce the false positive rate. Shang *et al*. [34] used steerable filters to guide the edge orientation of the lane boundary markers and introduced a VP guided Parallel Hough transform which showed improved performance under complex lighting and weather conditions. To sum up the handcrafted features and model based lane boundary extraction, Narote *et al*. [30] compiled a very recent and a comprehensive literature review. The prime limitations in these algorithms remain illumination invariance and occlusion.

With the advent of deep learning for image classification [23] the landscape of algorithms quickly changed and supervised CNNs snatched the top slots in tasks related to image classification [23], semantic segmentation [27] and object detection [12]. Lane detection, similarly, also witnessed a significant improvement with deep CNNs [35, 10, 22, 7, 28]. Pan *et al*. [33] used Spatial CNNs to detect lane boundaries as line segments. They employ a cascade of convolutional kernels across the height, width as well as the depth of the input images. Using a sequential propagation scheme to integrate information from every pixel, they reduce the memory and computational burden in contrast to dense Markov Random Field (MRFs), Conditional Random Fields (CRFs) and Recurrent Neural Networks (RNNs). The output is a class specific probability map of lane boundaries which are then sampled horizontally every 20 rows to produce a set of points. The cubic splines are fitted on sampled points to estimate lane boundary curves. Their algorithm is the top performing one at TuSimple benchmark [36] (accuracy 96.53%). They also introduced their dataset (CULane) and reported 71.6% F1 measure on its test set.

Lee *et al*. [24] used a Vanishing Point Guided network (VPGNet) by annotating VP in their training data. The VP constraint improves the lane detection as previously found by Changzheng *et al*. [6] but their data is not publicly available. Recent works, such as LaneNet by Neven *et al*. [31] introduced a CNN for lane boundary segmentation. They cluster class-specific semantic segmentation of the lane boundaries into the available number of lanes. In contrast to previous approaches such as [5, 24] that estimate IPM with the assumption of a fixed camera pose, Neven *et al*. learned a small network, HNet, to estimate it at run time. On TuSimple dataset their algorithm performs almost as good as VPGNet (accuracy 96.4%) but its performance is limited by the fact that deep learning approaches have inherent dependence on training data domain, therefore, change in camera hardware, pose and scene will greatly influence prediction of HNet and hence the outcome.

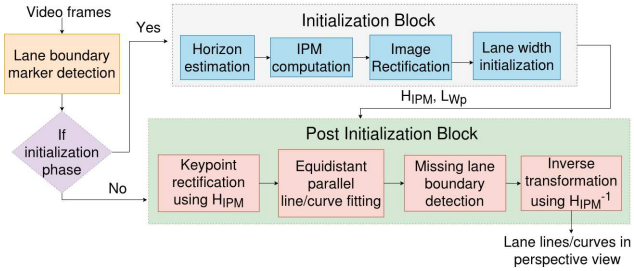


Figure 3. The block diagram of our end2end lane detection algorithm is shown here.

Deep learning is specifically good at learning context and semantic information of objects in the scene. Lane regions occupy more area than lane boundaries and hence are semantically strong contenders. With state-of-the-art performance in segmentation through Fully Convolutional Netowrks (FCN) [27] and Segmentation Networks (SegNets) [4], lane region segmentation has also been applied, but with not much success. A lane occupies a large region in the image and identifying all the pixels correctly, especially the boundary pixels becomes challenging. Generally, it is difficult to weight the training loss such that it penalizes the network more if border pixels get misclassified as they are more crucial in defining the lane boundary. Therefore, Zhang *et al*. [40] integrated both lane region segmentation and boundary pixel detection into two separate networks and combined them into a multi-task loss. The algorithm exceeded the performance of [33] with a total F1 measure of 73.1% on CULane. Recently, Hou *et al*. introduced Self Attention Distillation networks for lane detection that vary in performance (6 to 74 fps) and accuracy depending on the network architecture [14].

3. Our Approach

We believe that lane boundary detection can be generalized easily on a rather small dataset as compared to lane region. Tying boundary detection to region segmentation is not much helpful as is obvious from the reported F1 scores (71.6%) of Pan *et al*. [33] who used pure lane boundary detection contrast to Zhang *et al*. [40] (F1 73.6%) who complemented lane boundary and region detection in a joint framework on CULane dataset. Therefore, we focused only on lane boundaries. As generally is the case with deep learning methods for lane detection that output lane boundary maps which are sampled to produce boundary markers, we instead take motivation from keypoint detection in Mask-RCNN [12] and predict these lane markers directly instead of sampling them from segmentation maps. The main contribution of this paper is the post processing pipeline, shown in Figure 3, following the lane boundary markers detection. We fit lines to predicted keypoints and initialize lane width and camera rotation using the equidis-

tant and parallelism properties of the road lanes. At inference, if we are able to fit two lane boundaries (lines/curves) jointly using the initialized lane width within a reasonable tolerance (30%) and parallelism, only then we predict any missing lane boundaries and handle multiple predictions through weighted averaging. This pipeline greatly improves the lane detection accuracy and qualitative results are shown in Figure 1.

3.1. Lane Boundary Keypoint Detection

We represent each lane boundary using a fixed number of keypoints. The accurate estimation of horizon (which is critical for determining the IPM) is dependent upon the correct localization of Vanishing Point (VP). Therefore, during training, we sample more markers near the VP. Another reason for this is that points near the VP represent larger distance in the real world, so accurate localization of these points is crucial for precise line/curve fitting. The selected distribution of these markers along lane boundary on a sample TuSimple image is shown in Figure 2(b). To sample keypoints, we fit a cubic spline curve on the ground truth points for each lane boundary and then divide it vertically into three equal segments, where 50% of the total markers are sampled from a segment that is closer to the VP and $1/4^{th}$ of the total markers from the rest of the two segments each.

Inspired by the well known image segmentation networks (i.e. FCN [27] and SegNets [4]), our lane detection network is fully convolutional and consists of two blocks: an encoder block, responsible to encode all relevant low and high level features from the input image, and a decoder block that learns the relationship between provided low and high level features and makes the decision on top of it. The architecture diagram of our keypoint detector is shown in Figure 2(a). We used ResNet50 architecture [13] as backbone for information encoding while our decoder network consists of four convolution layers, a single transpose convolution layer, that upsamples the feature map by $4\times$ for precise localization of lane markers, and a final output convolution layer. In the output layer, we generate a one-hot mask for each marker just like Mask-RCNN, where only the marker's pixel is marked as foreground or one. To avoid inter-class competition among multiple keypoints, a per-mask cross-entropy loss is computed. The total loss of this network is the average of the loss for all K keypoints. At inference time, unique marker in each output map is computed by selecting the one with the highest score.

3.2. Camera Initialization

Camera initialization involves using camera intrinsics and road geometry to find Inverse Perspective Mapping (IPM) to synthetically rotate the viewpoint and lane width to fit equidistant parallel lanes in rectified view after the initialization.

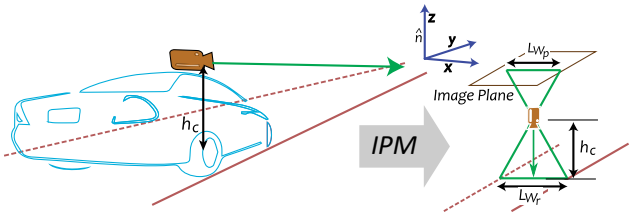


Figure 4. **(Left)** Perspective lane image demonstrating the right-hand camera system **(Right)** rectified camera view w.r.t road surface using IPM. h_c is the camera height, L_{Wr} is the lane width in real world in meters while L_{Wp} is the lane width on image plane in pixels given by Equation 1.

Our approach is aimed at a forward looking dashcam installed inside the driver's cabin. Configuration is done in a right-handed frame of reference. The road plane normal is along the Z-axis (upwards), Y-axis is in the driving direction and X-axis towards the right side of the vehicle. As shown in Figure 4(a), the camera is expected to have some tilt w.r.t X-axis whereas the pan angle w.r.t Y-axis is negligible and therefore, the image mainly has perspective distortion [25].

IPM is used to remove the perspective distortion in an image. Application of the IPM synthetically rotates the camera into a top (bird's eye) view thus rendering the intersecting lines as parallel; the way they actually are in the real world. The use of IPM improves the curve fitting as explained in earlier works such as [5, 24]. Recently, Neven *et al*. [31] trained a small CNN (HNet) to estimate IPM but its performance is limited by the fact that deep learning approaches have inherent dependence on training data domain, therefore, change in camera hardware, pose and scene will greatly influence the outcome. On the other hand, for a geometric approach, knowing camera pose could be challenging as it may vary from time to time. Therefore, we introduce a novel, accurate and robust way of estimating an IPM without any prior knowledge of the camera pose. We instead use road geometry to find camera pose and in-turn find the IPM. The following subsections explain all modules in the *initialization block* of the Figure 3.

3.2.1 Horizon Estimation

The estimation of IPM assumes a planar road scene having no elevation and more than one lanes (3 or more lane boundaries), such as a typical highway scenario. We start by fitting lines to the detected markers. A best fit line must have a minimum number of inlier markers (i.e., 25% of total number of markers for each lane boundary). Inlier markers are those having horizontal distance of under 8 pixels to the best 1st degree line fit through RANSAC [8].

Critical part of initialization is the estimation of a vanishing line (horizon) from these lines. This requires two sets of vanishing points (VPs), forward and lateral VPs. Forward

VPs can be found from the cross product of the world parallel lines whereas the process of finding lateral vanishing point is a bit involved. For that we use cross-ratios of three real-world parallel lines which are the lane lines as used by Ali *et al*. [2]. Once a set of forward and lateral VPs is found, we use their horizon estimation algorithm to find the best fitting horizon l_h.

3.2.2 IPM and Image Rectification

After estimating horizon, we compute plane normal $\hat{n}$ using camera intrinsic matrix *K* and horizon l_h [11] by $\hat{n} = K^T\, l_h$. The IPM is then computed using $H_{IPM} = KRK^{-1}$, where $R = [l_h \times \hat{n};\ (l_h \times \hat{n}) \times \hat{-n};\ \hat{-n}]$ is the rotation matrix and H_{IPM} is the rectification homography which will rotate the camera view to align its Z-axis with the road's normal.

Estimation of horizon and IPM on a short video clip completes the initialization of camera. Frames can then be simply rectified into a bird's eye view by applying this IPM (Figure 4). As long as the camera intrinsics or pose do not change, re-initialization is not required. Therefore, it is a one-off calculation for every camera setting.

3.2.3 Lane Width Initialization

A frame is considered an inlier frame if it has at least one forward and one lateral vanishing point. A pair of consecutive parallel lines are selected and the distance between the lines is calculated per frame through:

$$L_{Wp} = \frac{\sum_f \frac{|c_2 - c_1|}{\sqrt{a^2+b^2}}}{f}. \tag{1}$$

where f is the number of inlier frames and L_{Wp} is the average initialized lane width in pixels.

In real world, standard lane width typically vary across countries and is usually decided by state or federal department of transportation. Lane widths are commonly narrower on low volume roads and wider on higher volume roads and depends on the assumed maximum vehicle width. The Interstate Highway standards for the U.S. Interstate Highway System use a 3.7 meter standard lane width while in Europe, the minimum widths of lanes are generally between 2.5 to 3.25 meters [39]. Hence the ratios of these lanes widths vary within nearly 0.7 to 1.3. Therefore, an empirically selected threshold on lane width ratio (initialized:detected) $L_{WR} = 0.3$ should suffice.

3.3. Post-Initialization

In the post initialization phase, we use the computed homography and estimated lane width to rectify keypoints and fit equidistant parallel lines/curves on them. Following subsections explain all the modules shown in post-initialization block of Figure 3.

3.3.1 Equidistant Parallel Line/Curve Fitting

The lane boundary markers detected by CNN are rectified using the H_{IPM} found in Section 3.2.2. Second degree quadratic curves are fitted to randomly selected subset of markers. Three constraints are imposed on the curves in order to ensure correctness. First, the number of inliers, which have to be more than a minimum numbers (generally 25% of the total number of markers for lines and 50% for curves). For this, the perpendicular distance between the curve and a given marker is used as a measure. This process is repeated until the number of inliers cross a threshold or maximum iterations are reached. If it fails, then straight lines are tried.

After line/curve fitting, the second and the third constraints, parallelism and equidistant nature of lane lines, are ensured. If the two lines are straight, the parallelism is estimated from the difference between their slopes and the distance between them is computed using the Equation 1. In case of curved lane boundaries, the difference between the slopes of their tangents is used to judge parallelism and the lane width is calculated by intersecting a line perpendicular to the tangent of first curve to a point at the second curve and then finding the distance between the two points.

In its current form, our lane detector is trained for four lane boundaries (three lanes, i.e. left, center and right). If four lane boundaries do not satisfy all the three constraints, three lane boundaries are fitted and then two. At less than two lines/curves, the frame is ignored. Algorithm 1 explains this process for two lane boundaries.

3.3.2 Detecting Weak or Missing Lane Boundaries

In order to estimate the missing lanes, we used the initialized lane width in pixels (L_{Wp}). We calculate the lane boundary offset for the missing or low confidence lanes from Equation 1 as:

$$c_o = L_{Wp}\sqrt{a^2+b^2} \tag{2}$$

where c_o is the required offset in x-intercept for the missing lines. Adding c_o to the x-intercept of the detected lane boundaries in the rectified view, we predict the missing lane boundaries. For example, for a detected first lane's first boundary, we predict the number of missing lane boundaries on each side, i.e. $x = -\frac{b}{a}y + (c + c_o)$, $x = -\frac{b}{a}y + (c - c_o)$ (see Figure 5 (a)) .

After the missing lane boundary prediction, we perform a reciprocal weighted average of their x-coordinates near the forward VP and bottom of the image. If there are two predicted boundaries on the right side of a initial lane boundary as shown in Figure 5 (a), we use:

$$x'_1 = \frac{(w'x_1 + wx'_1)}{w + w'},\ \ x'_2 = \frac{(w'x_2 + wx'_2)}{w + w'} \tag{3}$$

where the weights w, w', w'' are $\{0,1,2\}$ respectively and are selected, based on the distance of the predicted

Algorithm 1: Lane boundary detection algorithm

Input: Detected markers from lane detector for a pair of consecutive lanes boundaries, fit2
Output: Detected lane boundaries

initialize;
$inliers_{tot} = 0$
Fit lines to lane boundaries:
Set M = min num of inliers;
while $iter \leq MAX\_ITER$ **do**
  Set $inliers_1 = 0$;
  Set $inliers_2 = 0$;
  Rectify lane markers using H_{IPM};
  Randomly sample n markers per line;
  if $fit2$==*1* **then**
    Fit 2^{nd} degree curve (Section 3.3.1);
    else
      Fit line;
    end
  end
  Count $inliesr_1$ and $inliers_2$ (Section 3.3.1);
  Check Constraint1:
  if $inliers_1 \geq M$ & $inliers_2 \geq M$ **then**
    Get slopes $m_1 = |a_1/b_1|, m_2 = |a_2/b_2|$ (from Equation $a_n x + b_n y + c_n = 0$);
    Get detected lane width L_{WD}(Section 3.2.3);
    Read Initialized lane width L_{Wp} (Equation 1);
    Check Constraints 2 and 3:
    if $|m_1 - m_2| \leq 0.1$ & $|1 - \frac{L_{WD}}{L_{Wp}}| \leq L_{WR}$ & $inliers^{n}_{tot} > inliers^{n-1}_{tot}$ **then**
      Set $lane\_detected$ = True;
      Update $inliers_{tot} = inliers_1 + inliers_2$;
    end
  end
end

lane boundary from detected lane boundary. Predicted lane boundary that is one lane width distance away is assigned a weight of 1, detected lane boundary is given a weight of 0 and so on. This enables robust retrieval of missing lane boundaries.

4. Evaluations and Discussion

We evaluated our algorithm on a number of publicly available lane datasets, namely, CULane [33], TuSimple [36] and Caltech [3]. In order to match lane boundaries with the ground truth we followed the TuSimple's criteria [1] for all datasets which allows prediction of up to two extra lane boundaries. The computational times listed in Table 1 were noted with GPU: GTX-1080 Ti 11 GB RAM and CPU: Core i7 6900K 3.20 GHz, except for the SAD-Nets [14] which where computed on GTX Titan X GPU by their authors.

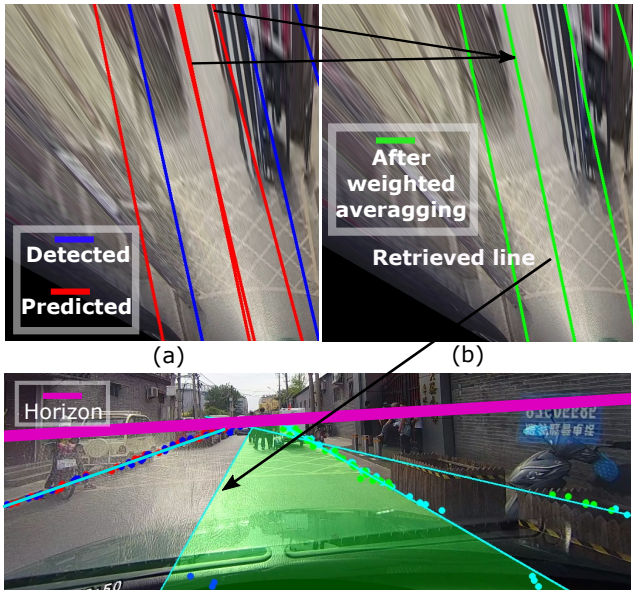


Figure 5. Detecting missing lane boundries. (a) Rectified view. Lane boundaries are predicted in red using c_o from section 3.3.2. (b) Filtered lane boundaries after weighted averaging (c) Recovered perspective view with all the four lane boundaries.

4.1. Results on CULane

CULane dataset consists of 55 hours of traffic video providing a total of 133,235 images. The training set contains 88,880 images, while the validation and test sets contains 9,675 and 34,680 each respectively. Depending on the scene categories, the test set is divided into eight subsets. Our lane marker detector was trained on the training set. We created lane boundaries in prediction and ground truth masks each of 30 pixels as used by [33] and [40], unless stated otherwise. Our results on the CULane test set and their comparison is given in Table 1.

As can be observed in the table, our method $Ours_{50}$ outperformed the state-of-the-art in lane boundary detection i.e. JLBNet [40], SCNN [33] and SAD-Nets [14] by an order of magnitude. In the *Normal, Crowded*, *Night*, *Arrow* and *Dazzle Light* categories, we had an improvement in F1 scores from 4 to 12% while in the *No Line* and *Shadow* categories, there had been an increase of almost 25% and 17% respectively. One odd case though was the *Curve* category in which we performed slightly better than SCNN but 3% lower than JLBNet and 5 to 6% lower than SAD-Nets. We will discuss the cause in the ablation studies. But we were able to improve the overall performance from 73.1% [40] and 72% SCNN [33] to almost 81%.

4.1.1 Ablation Studies

For the sake of ablation studies, we modified the lane boundary keypoint detector's architecture by changing the

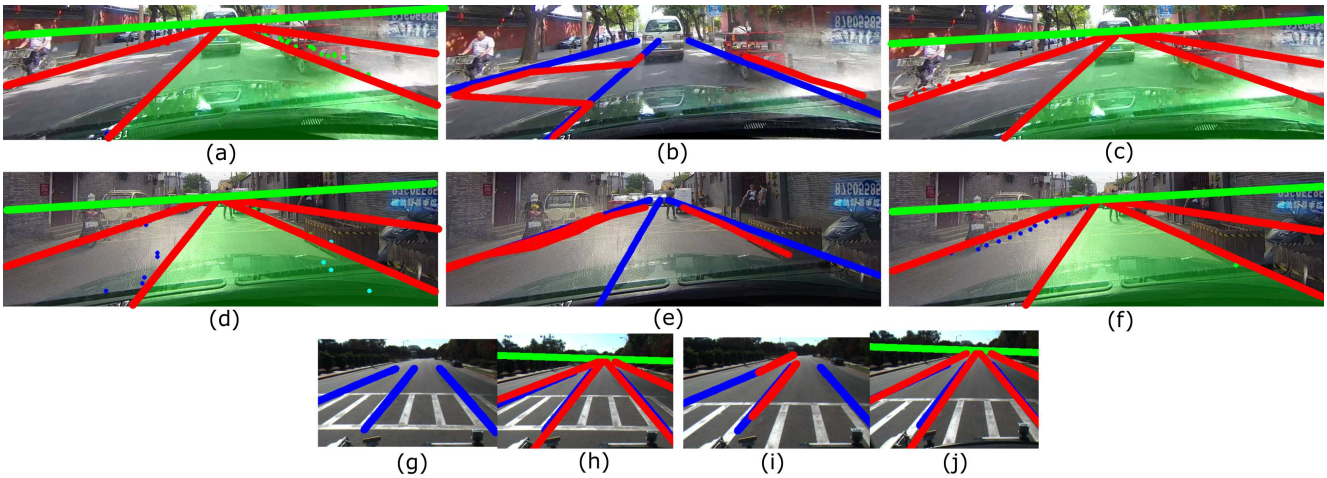


Figure 6. Results comparison (CULane samples: a to f, CalTech samples g to j). Detections in red and ground truth in blue and horizon line in green (a,d) Ours$_{40}$ (b,e) SCNN (c,f) SCNN lane boundary map + Ours$_{pp}$ post processing. (g,i) SCNN trained on CULane tested on Caltech. (h,j) Ours$_{40}$ trained on CULane and tested on CalTech.

Method	Normal	Crowded	Night	No Line	Shadow	Arrow	Dazzle Light	Curve	Total	FPS
SCNN[33]	0.883	0.753	0.686	0.365	0.593	0.821	0.531	0.594	0.72	9
JLBNet[40]	0.897	0.765	0.687	0.351	0.655	0.822	0.674	0.632	0.731	—
ENet-SAD[14]	0.901	0.688	0.660	0.416	0.659	0.840	0.602	0.657	0.708	74
R-18-SAD[14]	0.898	0.681	0.642	0.425	0.675	0.839	0.598	0.655	0.705	40
R-34-SAD[14]	0.899	0.685	0.646	0.422	0.677	0.838	0.599	**0.660**	0.707	20
R-101-SAD[14]	0.907	0.700	0.663	0.435	0.67	0.844	0.599	0.657	0.718	6
Ours$_{20}$	0.925	0.790	0.753	0.564	0.782	0.898	0.738	0.572	0.796	4.8
Ours$_{30}$	0.935	0.812	0.760	0.586	0.822	0.910	0.746	0.587	0.804	4.5
Ours$_{40}$	0.934	0.808	0.762	0.586	0.794	0.911	0.745	0.604	0.806	7.8
Ours$_{40_{c4}}$	0.931	0.801	0.759	0.581	0.799	0.911	0.747	0.573	0.802	4.2
Ours$_{50}$	**0.936**	**0.815**	**0.769**	**0.592**	**0.826**	**0.915**	**0.751**	0.607	**0.809**	3.7
SCNN$_{0.3,18}$ + Ours$_{pp}$	0.898	0.736	0.722	0.395	0.645	0.839	0.626	0.476	0.7500	4.3
Diff. with SCNN (%)	+1.48	-1.71	+3.58	+3.0	+5.19	+1.76	+9.54	-11.79	+3.09	–
SCNN$_{0.2,40}$ + Ours$_{pp}$	0.930	0.797	**0.775**	0.556	0.803	0.902	0.735	0.572	**0.808**	3.8
Diff. with SCNN (%)	+4.73	+4.38	+8.9	+19.11	+21.03	+8.13	+20.4	-2.25	+8.94	–

Table 1. F1 score for lane boundary detection on CULane [33]. Both Ground Truth (GT) and prediction were mapped to 30 pixel wide masks with IoU threshold of 0.5 or higher for True Positives. The lane detector used ResNet50 backbone. Ours$_{20}$: Conv$_5$ $\rightarrow$ Deconv block and 20 markers/boundary. Ours$_{30}$: Conv$_5$ $\rightarrow$ Deconv block and 30 markers/boundary and so on. Ours$_{40_{c4}}$: Conv$_4$ $\rightarrow$ Deconv block and 40 markers/boundary. F1 scores of SCNN and JLBNet were copied from JLBNet's paper [40] while for SAD-Nets from their paper [14].

total number of markers/boundary as well as the depth of the backbone to observe the effect on the lane detection results. We varied the number of markers/boundary to 20, 30 and 40 as well. Although the best results were achieved by $Ours_{50}$ (subscript represent the number of markers/boundary) as shown in Table 1, $Ours_{40}$ and $Ours_{30}$ also closely follow. These results show that reduction or increment in the number of points within the range 30 to 50 had no significant effect on the F1 scores but just a subtle improvement is observed with increasing the number of markers. We also reduced the depth of backbone and attached the decoder after the fourth block of ResNet architecture instead of the last block. This reduced the efficiency due to the removal of one max-pooling layer which increased the size of the output feature map while also marginally reducing the F1 scores. $Ours_{20}$ showed some reduction in overall F1 score (still higher than SCNN, JLB and SAD-Nets). This implies that with reduction in the number of markers below 30, our lane detection algorithm started to struggle. $Ours_{30}$ has the best trade-off between accuracy and efficiency. Please note that our algorithm was not optimized for efficiency and therefore, the FPS in the table is primarily for reference.

SCNN with Our Lane Prediction Algorithm

We also experimented by replacing lane detector (lane boundary marker detection block in Figure 3) in our pipeline with SCNN model [33] shared by the authors. The output lane boundary masks were sampled using criteria mentioned in [33] i.e., 18 points from every boundary with confidence ≥ 0.3. Results are shown in the table as SCNN+Ours$_{pp=0.3,18}$. The results were a bit of a mixture. F1 score improved from +1.48% in *Normal* to +9.54% in *Dazzle Light*. While it reduced by -1.71% in *Crowded* and -11.79% in *Curve*. Due to unbalanced dataset, the influence of compromised categories was mitigated and an overall F1 of 75% was achieved which was an improvement of about 3% from solo SCNN.

We also customized the sampling criteria in SCNN and used 40 samples at a threshold ≥ 0.2. Results are reported in the table as SCNN+Ours$_{pp=0.2,40}$, where *No Line/Shadow/Dazzle Light* showed improvements between 19.11% to 20.4%. An example of general improvement can be seen in Figure 6 (b and e) where the second lane boundary from the left was not correctly detected and marked as FN with solo SCNN. Our post processing of the markers detected the line with evidence (c) and without evidence (f).

Although the overall F1 score reached close to our best of 80.9%, the *Curve* category still remained 2% below the score of solo SCNN. The problem with the *Curve* category was that there were primarily fewer images (*Curved*: 421 images and for example *Normal*: 9621) and therefore they had less representation in the training. Secondly, during marker detection, occlusions reduce the number of detected markers. Therefore, during the parallel lane boundary fitting (section 3.3.1), the combined inliers remain fewer and such cases were labeled as FNs.

4.2. Results on CalTech

CalTech data has four video sets with a total of 1224 images. It's an old database (2008) and due to inconsistent ground truth, we only test it for one lane, preferably ego-lane as the annotations are there for at least two lanes in worst cases. Ego-lanes were identified through the slopes of their boundaries or their tangents (in case of curves) as in a perspective view, they have the steepest slopes in any direction. Since only SCNN's model was available, we tested SCNN and Ours$_{40}$ both trained on CULane. We mapped ground truth and predicted lanes each to 15 pixels and then to 30 pixels for two different tests. Results are shown in Table 2 (Left). Once again, our algorithm out performs SCNN noticeably in both tests with F1 scores of 89.90% (15:15) and 96.8% (30:30) respectively. Some sample images for comparison are shown in Figure 6 (g to j).

Data	Algorithm	F1	Data	Algorithm	Acc.
CalTech	SCNN(30:30)	0.900	TuSimple	SCNN	0.967
	Ours$_{40}$(30:30)	**0.968**		ENet-SAD	0.966
	SCNN(15:15)	0.520		Ours$_{40}$	0.955
	Ours$_{40}$(15:15)	0.899		**Ours**$_{40-gtRef}$	**0.979**

Table 2. (Left) F1 score on CalTech data. GT and predicted boundaries were mapped to 15 pixel (15:15) and 30 pixel (30:30) wide masks (Right) TuSimple Accuracy. SCNN results copied from [33]. In TuSimple, for Ours$_{40-gtRef}$ only, pixels along the ground truth lines marked as background are also kept as background in predicted lines.

4.3. Results on TuSimple

TuSimple [36] data contains 3626 training and 2782 test images. We trained Our$_{40}$ on the training data and the results on the test set are shown in Table 2 (Right), where the results for SCNN were obtained from their paper [33]. As we can observe, $Ours_{40}$ accuracy (95.5 %) is 1% lower than SCNN (96.5%). We designed our framework to be occlusion robust, so that lane boundaries across the road users and obstacles can also be detected. This was the idea behind using lane boundary markers and line fitting as argued in section 3.1 followed by missing lane boundary prediction in section 3.3.2. But in the TuSimple's ground truth, lane boundary occlusions are marked as background producing a discontinued ground truth. If we also treat those pixels as background for prediction during evaluation (Ours$_{40-gtRef}$), then our accuracy reaches almost 98% surpassing SCNN and SAD-Nets by more than 1.5%.

5. Conclusion

In this paper we have presented a robust and efficient lane detection algorithm. We have shown that using equidistant parallel lines constraint on detected lane lines, retrieval of missing and occluded lanes can be greatly enhanced. The results on three different publicly available and diverse datasets show substantial improvement from state-of-the-art which is a proof in itself that assumption holds. Our approach is modular and the missing lane detection block can be appended to any lane boundary detection network by sampling markers from lane boundaries.

For future work, we want to address auxiliary and loading lanes where the assumption of parallelism does not apply. We also want to improve the detection of curved lane boundaries. In its current form, our research takes the state-of-the-art forward and highlights new ways of dealing with one of the most critical challenges in autonomous driving by unifying deep learning and geometric computer vision to build a system utilizing best of both the worlds.

References

[1] https://github.com/TuSimple/tusimple-benchmark/tree/master/doc/lane_detection. *TuSimple evaluation criteria*, 2017.

[2] A. Ali, A. Hassan, A. Rafaqat, H. Khan, W. Kazmi, and A. Zaheer. Real-time vehicle distance estimation using single view geometry. *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020.

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

[4] V. Badrinarayanan, A. Kendall, and R. Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, dec 2017.

[5] M. Bertozzi and A. Broggi. Real-time lane and obstacle detection on the GOLD system. In *Proceedings of Conference on Intelligent Vehicles*, pages 213–218, 1996.

[6] H. Changzheng, H. Jin, and Y. Chaochao. An efficient lane markings detection and tracking method based on vanishing point constraints. *35th Chinese Control Conference (CCC)*, pages 6999–7004, 2016.

[7] S. Chougule, N. Koznek, A. Ismail, G. Adam, V. Narayan, and M. Schulze. Reliable Multilane Detection and Classification by Utilizing CNN as a Regression Network. In L. Leal-Taixé and S. Roth, editors, *European Conference on Computer Vision Workshops*, pages 740–752, Cham, 2018. Springer International Publishing.

[8] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6):381–395, jun 1981.

[9] F. Ghallabi, F. Nashashibi, G. El-Haj-Shhade, and M.-A. Mittet. LIDAR-Based Lane Marking Detection For Vehicle Positioning in an HD Map. In *21th International Conference on Intelligent Transportation Systems (ITSC)*, 2018 IEEE 21th International Conference on Intelligent Transportation Systems (ITSC), Maui, Hawaii,, United States, nov 2018. IEEE.

[10] 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 *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, jun 2016.

[11] R. Hartley and A. Zisserman. More Single View Geometry. In *Multiple view geometry in computer vision*, chapter 8, pages 205,216–219. Cambridge Univ. Press, 2nd edition, 2003.

[12] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask R-CNN. *International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[14] Y. Hou, Z. Ma, C. Liu, and C. C. Loy. Learning Lightweight Lane Detection CNNs by Self Attention Distillation. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[15] P. Hough. Method and Means for Recognizing Complex Patterns. U.S. Patent 3.069.654, dec 1962.

[16] (https://comma.ai). Comma AI's Open Pilot: An open source driving agent. 2019.

[17] (https://waymo.com). Google's Waymo: A self-driving taxi service. 2019.

[18] (https://www.mobileye.com). Intel's Mobile eye: ADAS and self-driving systems. 2019.

[19] (https://www.tesla.com/autopilot). Tesla's Autopilot: Full self-driving hardware. 2019.

[20] C. R. Jung and C. R. Kelber. Lane following and lane departure using a linear-parabolic model. *Image and Vision Computing*, 23(13):1192–1202, 2005.

[21] S. Kammel and B. Pitzer. Lidar-based lane marker detection and mapping. *2008 IEEE Intelligent Vehicles Symposium*, pages 1137–1142, 2008.

[22] J. Kim and C. Park. End-To-End Ego Lane Estimation Based on Sequential Transfer Learning for Self-Driving Cars. *Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1194–1202, 2017.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[24] 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 *International Conference on Computer Vision (ICCV)*, pages 1965–1973. IEEE, 2017.

[25] C. Lin and M. Wang. A vision based top-view transformation model for a vehicle parking assistant. *MDPI Sensors Journal*, pages 4431–4446, 2012.

[26] P. Lindner, E. Richter, G. Wanielik, K. Takagi, and A. Isogai. Multi-channel lidar processing for lane detection and estimation. In *12th International Conference on Intelligent Transportation Systems*, pages 1–6, oct 2009.

[27] J. Long, E. Shelhamer, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[28] X. Ma, A. Ismail, A. Soni, M. E. Carazo, V. Narayan, and M. Schulze. An efficient encoder-decoder CNN architecture for reliable multilane detection in real time. *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1444–1451, 2018.

[29] J. B. McDonald and J. B. M. Donald. Application of the Hough Transform to Lane Detection and Following on High Speed Roads. In *Proceeding of Irish Signals and Systems Conference*, 2001.

[30] S. P. Narote, P. N. Bhujbal, A. S. Narote, and D. M. Dhane. A review of recent advances in lane detection and departure warning system. *Pattern Recognition*, 73:216–234, 2018.

[31] D. Neven, B. D. Brabandere, S. Georgoulis, M. Proesmans, and L. V. Gool. Towards End-to-End Lane Detection: an Instance Segmentation Approach. *IEEE Intelligent Vehicles Symposium (IV)*, 2018.

[32] Y. Otsuka, S. Muramatsu, H. Takenaga, Y. Kobayashi, and T. Monj. Multitype lane markers recognition using local

edge direction. In *Intelligent Vehicle Symposium, 2002. IEEE*, volume 2, pages 604–609 vol.2, jun 2002.

[33] X. Pan, J. Shi, P. Luo, X. Wang, and X. Tang. Spatial As Deep: Spatial CNN for Traffic Scene Understanding. In *AAAI Conference on Artificial Intelligence*, pages 7276–7283, 2018.

[34] E. Shang, J. Li, X. An, and H. He. A real-time lane departure warning system based on FPGA. In *14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1243–1248, oct 2011.

[35] P. Y. Shinzato, V. Grassi, F. S. Osório, and D. F. Wolf. Fast visual road recognition and horizon detection using multiple artificial neural networks. *Intelligent Vehicles Symposium*, pages 1090–1095, 2012.

[36] TuSimple. TuSimple Velocity Estimation Challenge. pages https://github.com/TuSimple/tusimple–benchmark/tre, 2017.

[37] T. Veit, J. Tarel, P. Nicolle, and P. Charbonnier. Evaluation of Road Marking Feature Extraction. In *2008 11th International IEEE Conference on Intelligent Transportation Systems*, pages 174–181, oct 2008.

[38] J.-G. Wang, C.-J. Lin, and S.-M. Chen. Applying fuzzy method to vision-based lane detection and departure warning system. *Expert Systems with Applications*, 37(1):113–126, 2010.

[39] Www.eurotestmobility.net/eurotest.php. EuroTest.

[40] J. Zhang, Y. Xu, B. Ni, and Z. Duan. Geometric Constrained Joint Lane Segmentation and Lane Boundary Detection. In *European Conference on Computer Vision*, pages 37–47, 2018.