Real-Time Lane Estimation Using Deep Features and Extra Trees Regression

Vijay John[1](✉), Zheng Liu[1], Chunzhao Guo[2], Seiichi Mita[1], and Kiyosumi Kidono[2]

[1] Toyota Technological Institute, Nagoya, Japan
vijayjohn@toyota-ti.ac.jp
[2] Toyota Central R&D Labs, Nagakute, Japan

Abstract. In this paper, we present a robust real-time lane estimation algorithm by adopting a learning framework using the convolutional neural network and extra trees. By utilising the learning framework, the proposed algorithm predicts the ego-lane location in the given image even under conditions of lane marker occlusion or absence. In the algorithm, the convolutional neural network is trained to extract robust features from the road images. While the extra trees regression model is trained to predict the ego-lane location from the extracted road features. The extra trees are trained with input-output pairs of road features and ego-lane image points. The ego-lane image points correspond to Bezier spline control points used to define the left and right lane markers of the ego-lane. We validate our proposed algorithm using the publicly available Caltech dataset and an acquired dataset. A comparative analysis with a baseline algorithms, shows that our algorithm reports better lane estimation accuracy, besides being robust to the occlusion and absence of lane markers. We report a computational time of 45 ms per frame. Finally, we report a detailed parameter analysis of our proposed algorithm.

Keywords: Convolutional neural network · Extra trees · Lane detection · Occlusion

1 Introduction

Urban environment driving often requires to driver to concentrate while driving, which results in the increase of driver stress and fatigue. To provide assistance to the drivers, in recent years, there has been significant interest in the development of advanced driver assistance systems (ADAS) and autonomous vehicles. Within ADAS, lane detection plays an important role, and is used for applications such as lane following, lane departure warning, collision warning and vehicle path planning. The standard lane detection algorithm relies on the robust estimation of visible lane markers from the camera image, using vision and image processing algorithms. But the robust estimation of lanes is not trivial. Some of the challenges include degraded or absence of lane markers, lane marker occlusion, lane marker appearance variations, illumination variations and adverse environmental conditions such as snow and rain. Examples of challenges in the lane detection algorithm are presented in Fig. 1. Researchers often employ constraints such as tracking [1], region-of-interest (ROI) based estimation [2] and prior lanes [3] to overcome these limitations. However, even the use of additional constraints has some limitations and does not completely solve the lane detection problem [4].

In this paper, we propose an alternative scheme to address these issues using a learning framework. The proposed learning framework models the relationship between the entire road image and annotated ego-lane markers for various scenarios. The modeled relationship is then used to predict the ego-lane location within a given test road image. The ego-lane image location is represented using annotated Bezier spline control points, defined for the left and right lane markers. Using training pairs of road images and annotated lane control points, the ego-lane location is estimated in two learning-based steps. In the first step, we utilise the convolutional neural network (CNN) to extract robust and discriminative features from the entire input road image. We train the CNN to perform feature extraction by fine-tuning the pre-trained Places-CNN model [5] using a dataset of road images. In the second step, the extra trees regression algorithm is used to model the relationship between the extracted CNN-based road features and their corresponding annotated ego-lane markers. The trained models are then used to estimate the ego lane location in real-time during the testing phase, by CNN-based feature extraction and extra trees-based ego-lane prediction. Our main contribution to the lane detection literature is the use of a CNN and extra trees-based learning framework to accurately predict the ego-lane location in an image even under conditions of lane occlusion and absence. We validate our proposed algorithm on the publicly available Caltech dataset [6] and an acquired dataset. As shown in our experimental results, the proposed algorithm demonstrates the ability to accurately estimate the ego-lane, besides being robust to occlusion and absence of lane markers. Moreover, we report a real-time computational time of 45 ms per frame. A comparative analysis with the baseline algorithms is shown, along with a detailed parameter analysis. The rest of the paper is structured as follows. In Sect. 2 we report the literature review. The algorithm is presented in Sect. 3, and the experimental results are presented in Sect. 4. Finally, we summarize and conclude the paper in Sect. 5.

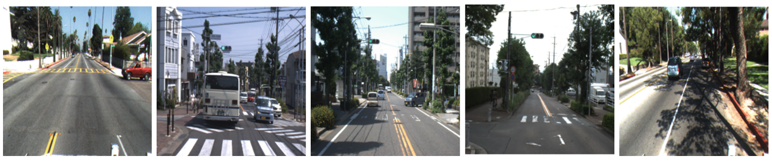


Fig. 1. An illustration of the various challenges in the lane detection including the absence of lane markers, degraded lane markers, shadows and occlusions.

2 Literature Review

Lane detection being an important problem in ADAS, has received significant attention in the research community. The summary of lane detection algorithms can be found in the surveys by Hillel et al. [4] and Yenikaya et al. [7]. The literature for the lane detection problem can be categorized into feature extraction techniques [1,2] and model-based techniques [3,8]. In feature extraction techniques, the lane markers are identified from the input image using image processing and fitting algorithms. On the other hand, the model-based techniques are generative models, where candidate lane models are evaluated for optimum fitness with the image-based features. We next present a brief overview of the literature in both the techniques.

In the standard feature extraction technique, the road image is first preprocessed to remove noise. Next, using image thresholding techniques, candidate lane markers are localised in the image. The final lanes are then identified from the candidate set. For example, Andrew et al. [2] localise the lanes in the image by identifying the painted lines on the road, and prune the candidates using orientation and length-based threshold. To enhance the accuracy of feature extraction-based techniques, researchers utilise techniques such as spatial constraints (ROI), tracking and inverse perspective transformations (IPT) into their algorithms. The ROI is used to constrain the lane search to a specific region in the image. While the tracking algorithm is used to incorporate temporal information to enhance the detection accuracy. Arshad et al. [9] utilise an adaptive ROI within their colour-threshold-based lane detection algorithm to identify the lanes. While Choi et al. [1] propose a lane detection algorithm using template matching, RANSAC and Kalman filtering. The template matching framework is used to obtain candidate lanes from the image, which are then pruned using the RANSAC and Kalman filter. In the work by Aly et al. [6], the IPT algorithm is used to generate the top view of the road, which are then filtered using oriented Gaussian filters to obtain candidate lanes. The candidate lanes are pruned using RANSAC and Bezier spline fitting algorithm. In the recent work by Kim et al. [10] a RANSAC and CNN is used to estimate the lanes from the edges in the image. The CNN is trained to generate output binary lane maps from input edge maps containing visible lane markers. The output binary lane maps are then used within the RANSAC algorithm as priors to estimate the lanes. However, the CNN-based output binary lane maps are dependent on the presence of visible lane markers in the input edge maps. Summarising the feature extraction methods, we observe that these methods are simple and easy to implement, but are limited by their dependence on strong lane cues.

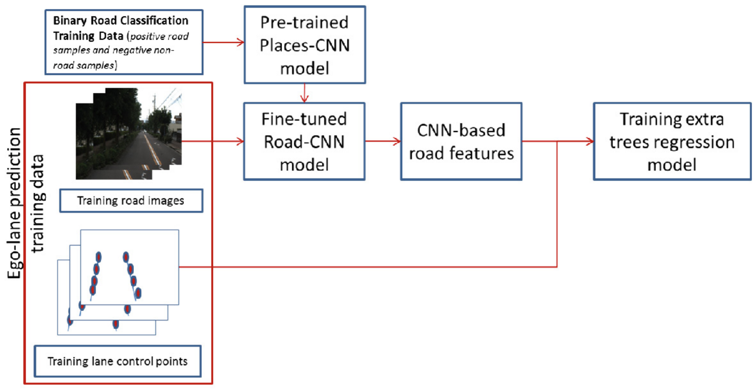
In the model-based technique, lane models defined using parameters are used within a model-fitting framework to estimate the lanes. Candidate lane models generated using parameter estimates are evaluated within model-fitting frameworks such as RANSAC, particle filtering, etc. For example, Sehestedt et al. [8] evaluate the candidate lanes using the particle filter in the IPT edge image. In the work by Huang et al. [11], basis curves are used within a probabilistic framework to estimate the lanes. Lane estimates defined by the basis curves, are evaluated with observations from the road in a probabilistic framework. To enhance the accuracy of the lane estimation, the authors integrate the curb detection and road paint detection algorithm in their framework. Similar to the feature extraction techniques, researchers also employ constraints within the model-based framework. For example, Kowasri et al. [3] constrain the particle filters-based model fitting algorithm using lane priors. The lane priors are annotated on the map, and are retrieved by the vehicle during testing. Comparing the two categories of literature, we can observe that the model-based technique are more robust. But the performance of both these methods are affected by degraded lane markers, absence of lane markers, lane occlusion, and adverse weather conditions such as snow and rain. Unlike the aforementioned approaches, in this paper, we propose to address these issues using a learning-based approach by modeling the relationship between road image and annotated lane markers. The modeled relationship is then used to predict ego-lanes for various challenging scenarios.

3 Algorithm

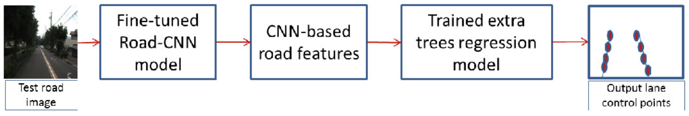
In this paper, we formulate the lane detection problem using two learning algorithms, the CNN and the extra trees. The proposed algorithm consists of a training and testing phase. During the training phase, the CNN is trained to extract D-dim road features $\mathbf{f} \in \mathbb{R}^D$ from the input road image I. Next, the extra trees model is trained to predict the ego-lane in the image using training pairs of input road features $F = \{\mathbf{f}_n\}_{n=1}^N$ and output Bezier splines $\mathbf{B} = \{B_n\}_{n=1}^N$. In the training output, $B_n = [\mathbf{b}_n^l, \mathbf{b}_n^r]$, represents the set of Bezier splines for the n-th image, where $\mathbf{b}_n^l$ and $\mathbf{b}_n^r$ corresponds to the Bezier spline control points for the left and right lane marker. Each Bezier spline is defined using 4 control points or 8 image pixel coordinates. The 8 image pixel coordinates are manually annotated within a predefined ROI in the image, and used for training and testing. In the testing phase, for a given input road image I_t the trained CNN extracts the features $\mathbf{f}_t$. The feature $\mathbf{f}_t$ is then given as an input to the trained extra trees, which subsequently predicts the ego-lane using the output Bezier splines B_t. An overview of our proposed algorithm is shown in Fig. 2. We next present a detailed overview of the training and testing phase.

3.1 Training

Convolutional Neural Network. The convolutional neural network (CNN) is a deep learning framework used for multi-class classification [5]. It has reported state-of-the-art detection and classification result in various image classification, speech recognition applications [5]. The deep learning framework is an end-to-end machine learning framework, where multiple layers of processing are used to simultaneously perform feature extraction and feature classification. To perform the feature extraction and classification, CNN based on the multi-layered perceptron learns the weights and bias in each layer. The lower layers of CNN extract the low level features, while the higher layers learn the high-level features.


Binary Road Classification Training Data (positive road samples and negative non-road samples)
Pre-trained Places-CNN model
Fine-tuned Road-CNN model
CNN-based road features
Training extra trees regression model
Ego-lane prediction training data
Training road images
Training lane control points

(a)


Test road image
Fine-tuned Road-CNN model
CNN-based road features
Trained extra trees regression model
Output lane control points

(b)

Fig. 2. An overview of the (a) training and (b) testing phase of the proposed algorithm

The extracted features are represented as feature maps in the CNN network. Given the feature maps, CNN performs the feature classification inherently in the final layers. The CNN architecture is represented using convolutional layers, pooling layers, fully connected layers, drop-out layers and loss layers. The convolutional layers and pooling layers are used in the initial layers to inherently perform feature extraction. The final layers contain the fully connected layers, drop-out layers and loss layers function, which are used to the perform inherent feature classification. To update the weights in the different layers, the stochastic gradient descent is used, where the weights are updated by the following equations.

$$V_{t+1} = \mu V_t - \alpha \nabla L(W_t) \tag{1}$$

$$W_{t+1} = W_t + V_{t+1} \tag{2}$$

where W_t is the previous weight and V_t is the previous weight update at iteration t. V_{t+1} and W_{t+1} are the updated weight update and weights at $t+1$. $\nabla L(W_t)$ is the negative gradient of the previous weight. α is the learning rate for the weight of the negative gradient. μ is the momentum or the weight of the previous update. Additionally, each layer has a learning rate multiplier term for the weights and bias β_w and β_b, respectively. This is used to vary the α during training for each layer to facilitate faster convergence.

In this paper, the CNN is utilised to extract the features from the entire road image. To extract the road specific features, the weights (filters) and bias of the CNN model pre-trained on the *Places* dataset [5] is fine-tuned with a dataset of road images. The *Places* dataset is a scene-centric database with 205 scene classes and 2.5 million images used for scene recognition. Consequently, the Places-CNN model filters are highly tuned for scene feature extraction and classification. The detailed architecture of the pre-trained Places-CNN model is as follows: The size of the input RGB is $256 \times 256 \times 3$, and the first convolutional layer consists of 96 filters with size 11×11 with stride 4. The second convolutional layer contains 256 filters of size 5×5. Both the first and second convolutional layers are followed by maximum pooling with filter 3×3 and stride 2 and local response normalisation over a 5×5 window. The third and fourth convolutional layer contains 384 filters of size 3×3. The fifth convolutional layer contains 256 filters of size 3×3, followed by maximum pooling with filter size 3×3. The next two layers are the fully connected layers with 4096 neurons followed by a drop-out layer with drop-out ratio of 0.5. Note that the convolutional and fully connected layer contain the relu activation function. The output layer contain 205 neurons with softmax functions. Finally, the α is set to 0.01, μ is set to 0.9 and $\beta_w = 1$ and $\beta_b = 2$ for all the layers.

To perform the fine-tuning for road feature extraction, we formulate the multiclass Places-CNN model as a binary road classifier. We adopt this formulation to fine-tune the feature extraction layers of the (C1–C5) through the backpropagation algorithm. To modify the Places-CNN model as a binary classifier, we replace the 205 neurons in the output layer with 2 neurons. Henceforth we refer to this modified binary model, as the road-CNN model. The fine-tuning for the road-CNN model is done using 100000 positive road scene samples and 100000 negative non-road scene samples. Note that the binary classification modification is primarily done to fine-tune the pre-trained feature extraction layers, owing to the CNN's characteristic of network training using backpropagation from the output layers.

During the fine-tuning step, the weights and bias of the road-CNN feature extraction layers, $C1$–$C5$, are initialised with the corresponding pre-trained weights and biases of the Places-CNN model. On the other hand, the weights and bias for the fully connected layers $FC6$–$FC8$ or feature classification layers are initialised without the pre-trained weights and bias. To reflect the pre-initialisation, we lower the learning rate α to a lower value 0.001, compared to the Places-CNN model. But, the momentum and learning multipliers (β_w and β_b) for convolutional layers $C1$–$C5$ are kept the same (0.9,1,2). On the other hand, to enable the feature extraction layers to learn quickly, β_w and β_b for fully connected layers $FC6$ and $FC7$ are set as 2 and 4, while the β_w and β_b for the output layer is set as 10 and 20. The number of training iterations is lowered to 50000. Using the above described parameters, we fine-tune the road-CNN and learn the road-specific weights and bias. The fine-tuned road-CNN is then used to extract the road features, which corresponds to the output feature maps of the fifth pooling layer ($P5$). Consequently, each road feature $\mathbf{f}$ is represented by

256 feature maps of size 7 × 7. Thus generating a 12544 dim feature vector. We refer to this feature vector as the *complete* feature vector. An illustration of the filters and feature maps of the road-CNN model is presented in Fig. 3

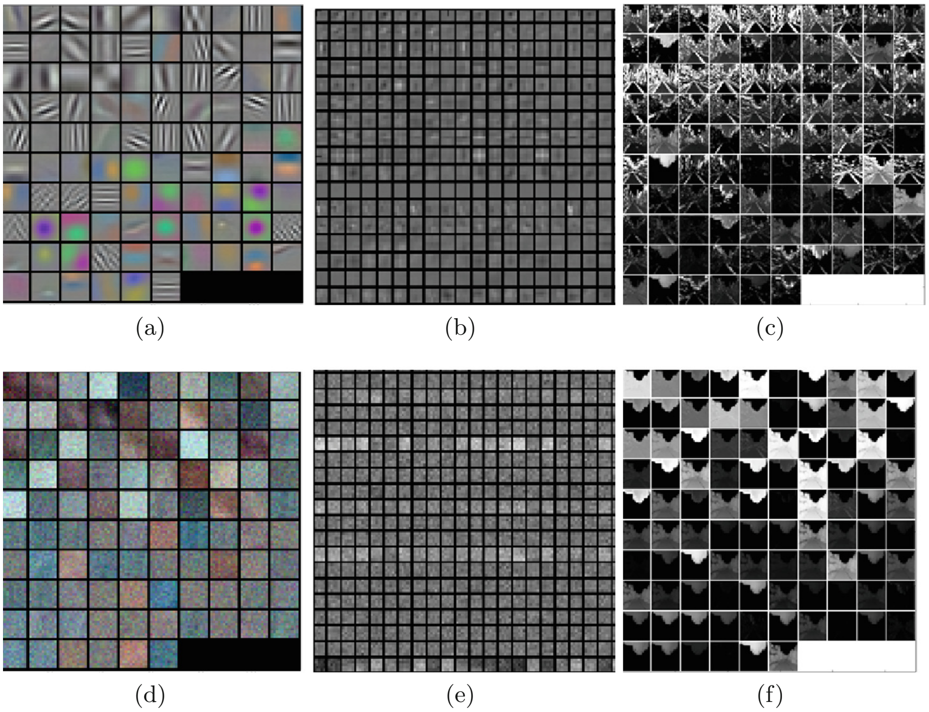


Fig. 3. A visualization of the (a) C1-filters, (b) C2-filters, and (c) C1-feature maps for the fine-tuned road-CNN. The (d) C1-filters, (e) C2-filters and (f) C1-feature maps for the directly-trained road-CNN (Details of this model given in Sect. 4.2)

Extra Trees Regression. The extra trees are an extension of the random forest regression model and proposed by Geurts et al. [12]. The extra trees belong to the class of decision tree-based ensemble learning methods. In decision tree-based ensemble methods, multiple decision trees are used to perform classification and regression tasks [13]. Random forest and extra trees are important algorithms within this class and have reported state-of-the-art performance on many regression tasks with high-dimensional input and outputs [13]. The random forests are trained to perform the regression tasks using the techniques of tree bagging and feature bagging. In tree bagging, each decision tree in the ensemble is trained on a random subset of the training data, generating an ensemble of decision trees. Given, the ensemble of decision trees, the feature bagging technique is used to perform the split at each node in the decision tree. The feature bagging-based split is performed in two steps. In the first step, random subset of features are selected from the previously selected training data subset. In the second step, the best subset feature and its corresponding value are chosen to perform the decision split. Typically, the best feature is selected based on the information gain or Gini criteria [12]. The extra trees was proposed as an computationally efficient and highly randomised extension of the random forest. There are two main differences between the extra trees and the random forest. Firstly, unlike the random forest, the extra trees does not use the tree bagging step to generate the training subset for each tree. The entire training set is used to train all the decision trees in the ensemble. Secondly, in the node-splitting step, the extra trees randomly selects the best feature along with the corresponding value to split the node. These two differences results in the extra trees being less susceptible to overfitting and reporting better performance [12]. In our problem, we train the extra trees regression model with 50 trees to predict the ego-lane location for a given input image. During training, the extra trees model is trained using pairs of input road features, $F = \{\mathbf{f}_n\}_{n=1}^{N}$, and annotated output Bezier spline control points, $\mathbf{B} = \{B_n\}_{n=1}^{N}$.

3.2 Testing

During the real-time testing phase, we use the trained CNN and extra trees model to predict the ego-lane location in an given input test image I_t, resized to $256 \times 256 \times 3$. First, the trained CNN is used to extract the *complete* feature maps or the road features $\mathbf{f}_t$. The road features are then given as an input to the trained extra trees, which outputs the ego-lane location in the image using the Bezier spline control points B_t.

4 Experimental Results

The proposed algorithm was validated with the publically available Caltech dataset [6] and an acquired dataset. We performed a comparative analysis with a baseline algorithm and report a better lane detection accuracy even under occlusion and absence of lane markers. Additionally, a detailed parameter analysis of our algorithm is performed. In the experiments, we validate our proposed method using the Caltech dataset and an acquired dataset. The Caltech dataset has 1224 images with annotated Bezier spline control points, while the acquired dataset contains 2279 frames with manual annotation. The training and testing of our proposed algorithm was done using a 5-fold cross validation scheme on both the datasets. The algorithm is implemented on a Linux machine with Nvidia Geforce GTX960 graphics card using Python and GPU-based Caffe software tool [14]. The scikit machine learning library was used to implement the extra trees.

4.1 Comparative Analysis

In this section, we report the results of a comparative analysis performed with two baseline algorithms. The first baseline algorithm is based on the Hough

Table 1. Det. Accuracy *Caltech*

Algorithm	Lanes	Det. rate
Proposed algo.	2450	98.9 %
ROI-Hough-Ransac.	2450	94.2 %
ROI-Hough.	2450	88.4 %

Table 2. Det. Accuracy *Acquired*

Algorithm	Lanes	Det. rate
Proposed algo.	4560	99.3 %
ROI-Hough-Ransac.	4560	96.8 %
ROI-Hough.	4560	92.6 %

transform-based lane detection framework [9]. In this algorithm, we first identify the lane ROI in the image, and utilise a intensity and colour-based threshold to identify the set of candidate lines. Following morphological operations, the candidate lanes are pruned using the Hough transform. Finally, the two strongest lanes are used to identify the ego-lane markers in the image. In the second baseline, based on the work by Aly et al. [6], we utilise edge information to identify candidate lanes within a lane ROI. The candidate lanes are further pruned by the Hough transform, before estimating the ego-lane using Ransac.

To compare the algorithms, we use their respective lane detection accuracies. An estimated lane marker is considered as a true positive, when the distance between the estimated lane marker points and the ground truth lane marker points is less than 20 pixels. Prior to calculating the Euclidean distance between the lane marker points, a greedy nearest neighbour search is performed to match the estimated spline points with their corresponding ground truth spline points. The lane marker points for the proposed algorithm are derived from the Bezier splines, which are in turn, interpolated from the predicted control points. A similar technique is adopted to extract the ground-truth lane marker points. On the other hand, the lane marker points of the baseline algorithm are directly generated from algorithm's output. Since both our proposed and baseline algorithms only estimates the ego-lane in the final output, every missed lane marker or false negative also corresponds to a false positive. Consequently, we do not report the false positive rate.

The lane detection accuracy between the proposed algorithm and the baseline algorithm is evaluated using the testing subset of each round of the 5-fold cross validation. As shown in Tables 1 and 2, it can be seen that the proposed algorithm reports better lane detection algorithm than the baseline algorithm. Additionally, as shown in Fig. 4, our proposed algorithm estimates the ego-lane for frames, where the lane markers are either occluded or absent. The baseline algorithm fails to detect the lane markers under these conditions. We report a computational time of 45 ms per frame using the GPU, where the CNN-based feature extraction takes an average 44 ms, while the extra trees regression takes 1 ms.

4.2 Parameter Analysis

Here we perform a detailed parameter analysis of our algorithm by reporting the lane detection accuracy. For the parameter analysis, we first evaluate the fine-tuning of the road-CNN model. Second, we evaluate the CNN-based features

Table 3. Det Acc. *Caltech*

Feat-Reg	Ext. trees	Rand forest	Lasso
Complete	98.9 %	95.9 %	95.3 %
HOG	92.9 %	90.3 %	65.1 %
SURF	93.2 %	90.4 %	65.2 %

Table 4. Det Acc. *Acquired*

Feat-Reg	Ext. trees	Rand forest	Lasso
Complete	99.3 %	95.6 %	94.8 %
HOG	91.8 %	90.7 %	76.8 %
SURF	92.6 %	91.4 %	76.7 %

with different feature descriptors. Third, we evaluate the features used as the regression input. Finally, we evaluate the regression models used for the lane prediction.

Fine-Tuned Feature Extraction. In this experiment, we validate the need to fine-tune the pre-trained Places-CNN. To perform the validation, we set-up a directly trained (DT) road-CNN model, where we train all the layers of the road-CNN model without any pre-initialisation using the pre-trained Places-CNN weights and bias. On the other hand, in the fine-tuned road-CNN model we pre-initialise the weights and bias for layers $C1 - C5$. The training for DT road-CNN models is done using thee same binary dataset (100000 positive and 100000 negative samples). Following the training, we evaluate the fine-tuned and directly-trained CNN models on a separate test dataset of 10000 road scenes and 10000 negative scenes and report the classification accuracies. Based on the experimental results, we observe that the fine-tuned road-CNN model achieves a 100 % classification accuracy, while the DT CNN only achieves a 75 % accuracy. This validates the need to adopt a pre-trained network for fine-tuning, instead of directly training the CNN. In Fig. 3, we present the filters and feature maps of the two CNN networks. It can be clearly seen that the fine-tuned filters and feature maps are well-trained, unlike the directly trained filters and features maps.

Different Features. To validate the lane detection accuracy due to the CNN-based road features, we train the regression model with the HOG and SURF-based road features [15] and report the lane detection accuracies. As shown in Tables 3 and 4, the performance of the CNN-based road features is much better than both the HOG and SURF-based features across different regression models.

CNN-Based Feature Descriptor. We perform an experiment by considering different feature descriptors to represent the CNN-based feature (*complete*). For this experiment, we consider three different feature descriptors with reduced dimensions, the *mean*, *phist* and *bhist*. In the *mean* descriptor the 256, 7×7, feature maps in the original feature vector are averaged, resulting in a 49-dim feature vector ($1 \times 7 \times 7$). In the *phist*, a 10-bin histogram is used to represent the distribution of values across the 256 feature maps at each 7×7 map index resulting in a 490-dim feature vector ($10 \times 7 \times 7$). Finally, in the *bhist* the 10-bin histogram is used to represent the feature distribution across 3 disjoint blocks of the 256 feature maps at each 7×7 index. This results in a 1470-dim feature vector ($30 \times 7 \times 7$). Based on our experimental results, we observe that none of

Table 5. Det. Acc *Caltech*

Feat-Reg	Ext. trees	Rand forest	Lasso
Complete	98.9 %	95.9 %	95.3 %
Mean	94.2 %	90.6 %	65 %
PHist	90.5 %	87.0 %	78.3 %
BHist	89.6 %	87.6 %	82 %

Table 6. Det. Acc *Acquired*

Feat-Reg	Ext. trees	Rand forest	Lasso
Complete	99.3 %	95.6 %	94.8 %
Mean	94.0 %	91.1 %	77.7 %
PHist	91.4 %	88.6 %	83.9 %
BHist	95.9 %	94.8 %	93.7 %

the feature descriptors demonstrate an improved lane detection accuracy, inspite of reducing the computational complexity of the regression model. This is shown in Tables 5 and 6.

Different Regression Models. The performance of the extra trees is evaluated in this section. Trained random forest and Lasso regression models are used to predict the lanes for different features and feature descriptors. As observed in all the tables above (Tables 1, 2, 3, 4, 5 and 6), the performance of the extra trees is better than the random forest and the Lasso models, across different features

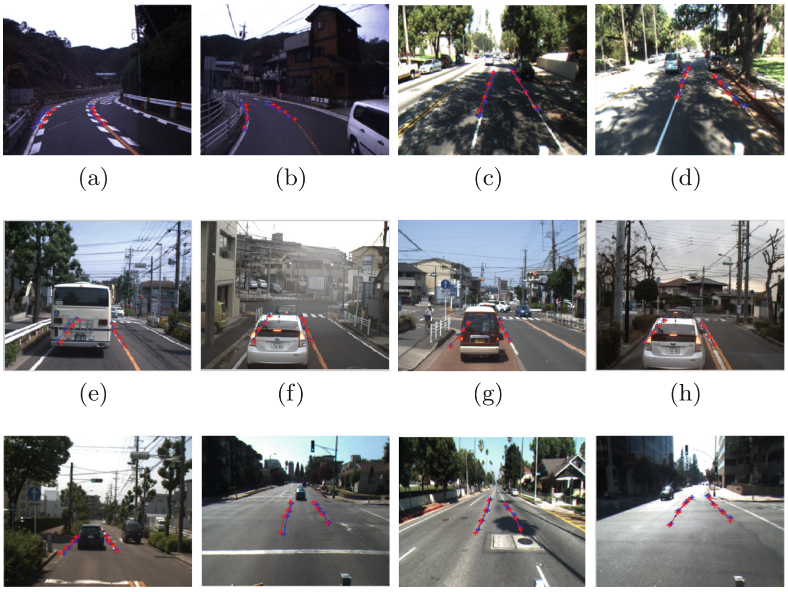


Fig. 4. Some lane detection examples of our proposed algorithm for challenges scenarios where the ego-lane markers are (a–b) curved, (c) shadowed, (d) shadowed and absent, (e–f) partially occluded, (g) partially occluded, absent with different coloured road surface, (h) occluded and absent, (i–l) absent. The blue splines are the ground truths, while the red splines are the predicted splines. The baseline algorithms fails to detect the ego-lane markers under these conditions.

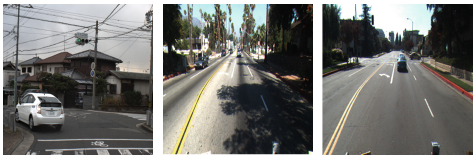


Fig. 5. Missed lane marker detections by our algorithm, for a hard left turn and suddenly emerging left lanes.

and feature descriptors. Additionally, we can also observe that the performance of the random forest is better than the Lasso model.

Discussion. Based on the experimental results, we observe that the proposed algorithm demonstrates better lane detection accuracy. Additionally, the advantage of the CNN-based road features and extra trees regression model is clearly demonstrated. The algorithm is robust to lane marker occlusions and absent lane markers. The few missed lane detections occur for lane marker scenes which occur rarely, resulting in limited training samples for the extra trees regression. Examples of missed lane detections classes are shown in Fig. 5. Note that this can be addressed by increasing the training samples for the particular class.

5 Conclusion and Future Work

In this paper, a real-time learning-based lane detection is proposed using the convolutional neural network and extra tree regression model. The proposed learning framework models the relationship between the input road image and annotated lane markers, which are then used to perform ego-lane prediction. In the learning framework, the pre-trained Places-convolutional neural network is fine-tuned to extract features from the input road images. These extracted features are used along with annotated ego-lane markers to train the extra trees regression model. The trained extra trees are then used to predict the ego-lanes in real-time, during the testing phase, using the convolutional neural network-based road features. We validate the proposed algorithm on the publicly available Caltech dataset and an acquired dataset, and perform a comparative analysis with baseline lane detection algorithms. Based on our experimental results, we report better lane detection accuracies. More importantly, our proposed approach is not affected by either occlusions or the absence of lane markers. A detailed parameter analysis of the proposed algorithm is also performed. In the future work, we will validate the algorithm on a larger dataset and extend the algorithm to identifying multiple lanes.

References

1. Choi, H., Park, J., Choi, W., Oh, S.: Vision-based fusion of robust lane tracking and forward vehicle detection in a real driving environment. Int. J. Automot. Technol. **13**(4), 653–669 (2012)
2. Andrew, H., Lai, S., Nelson, H., Yung, C.: Lane detection by orientation and length discrimination. IEEE Trans. Syst. Man Cybern. Part B **30**(4), 539–548 (2000)
3. Kowsari, T., Beauchemin, S.S., Bauer, M.A.: Map-based lane and obstacle-free area detection. In: VISAPP (2014)
4. Bar Hillel, A., Lerner, R., Levi, D., Raz, G.: Recent progress in road, lane detection: a survey. Mach. Vis. Appl. **25**(3), 727–745 (2014)
5. Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., Oliva, A.: Learning deep features for scene recognition using places database. In: Advances in Neural Information Processing Systems (2014)
6. Aly, M.: Real time detection of lane markers in urban streets. In: Proceedings of the Intelligent Vehicles Symposium (2008)
7. Yenikaya, S., Yenikaya, G., Düven, E.: Keeping the vehicle on the road: a survey on on-road lane detection systems. ACM Comput. Surv. **46**(1), 2:1–2:43 (2013)
8. Sehestedt, S., Kodagoda, S., Alempijevic, A., Dissanayake, G.: Efficient lane detection and tracking in urban environments. In: Proceedings of the European Conference on Mobile Robots (2007)
9. Arshad, N., Moon, K., Park, S., Kim, J.: Lane detection with moving vehicle using colour information. In: World Congress on Engineering and Computer Science (2011)
10. Kim, J., Lee, M.: Robust lane detection based on convolutional neural network and random sample consensus. In: Yap, K.S., Wong, K.W., Teoh, A., Huang, K., Loo, C.K. (eds.) ICONIP 2014, Part I. LNCS, vol. 8834, pp. 454–461. Springer, Heidelberg (2014)
11. Huang, A.S., Teller, S.: Probabilistic lane estimation for autonomous driving using basis curves. Auton. Robots **31**(2–3), 269–283 (2011)
12. Geurts, P., Ernst, D., Wehenkel, L.: Extremely randomized trees. Mach. Learn. **63**(1), 3–42 (2006)
13. Gall, J., Yao, A., Razavi, N., Van Gool, L., Lempitsky, V.: Hough forests for object detection, tracking, and action recognition. IEEE Trans. Pattern Anal. Mach. Intell. **33**(11), 2188–2202 (2011)
14. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guaddarrame, S., Darrel, T.: Caffe: convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093 (2014)
15. Song, J., Ma, Y., Hu, F., Lao, S., Zhao, Y.: Scalable image retrieval based on feature forest. In: Zha, H., Taniguchi, R., Maybank, S. (eds.) ACCV 2009, Part III. LNCS, vol. 5996, pp. 506–515. Springer, Heidelberg (2010)