Neurocomputing 280 (2018) 46–55

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Lane marking detection via deep convolutional neural network

Yan Tian[a], Judith Gelernter[b], Xun Wang[a,*], Weigang Chen[a], Junxiang Gao[c], Yujie Zhang[a], Xiaolan Li[a]

[a] School of Computer and Information Engineering, Zhejiang Gongshang University, Hangzhou, PR China
[b] Information Technology Laboratory, National Institute of Standards and Technology, Pittsburgh, US
[c] College of Science, Huazhong Agricultural University, Wuhan, PR China

ARTICLE INFO

Article history:
Received 30 January 2017
Revised 16 September 2017
Accepted 22 September 2017
Available online 21 November 2017

Keywords:
Lane marking detection
Intelligent transportation systems
Deep learning
Image processing
Computer vision

ABSTRACT

Research on Faster R-CNN has recently witnessed the progress in both accuracy and execution efficiency in detecting objects such as faces, hands or pedestrians in photograph or video. However, constrained by the size of its convolution feature map output, it is unable to clearly detect small or tiny objects. Therefore, we presented a fast, deep convolutional neural network based on a modified Faster R-CNN. Multiple strategies, such as fast multi-level combination, context cues, and a new anchor generating method were employed for small object detection in this paper. We demonstrated performance of our algorithm both on the KITTI-ROAD dataset and our own traffic scene lane markings dataset. Experiments demonstrated that our algorithm obtained better accuracy than Faster R-CNN in small object detection.

1. Introduction

Lane marking detection and localization in traffic scene images is crucial for Intelligent Transportation Systems, which can be used in Automatic Vehicle Driving and Advanced Driver Assistant System (ADAS). Numerous collision accidents are caused by at least one of the vehicles driving out of lane. If lane departure events are early discovered and corrected, some collisions could be avoided. This research improves upon a method to automatically recognize the lane markings on the road with a machine learning algorithm, and a driving out of lane equation is derived as well. The result produced by the equation determines if the driver should be warned about out of lane situation.

Since lane markings are line like features, a wide variety of traditional image processing algorithms could be applied for the task, such as edge detection [1], template matching [2] and Hough Transform [3], in which color, texture, edge and other low-level features are used to detect the area or the edge of the lanes in images. However, those traditional methods are not suitable for detecting lane markings due to (1) variations of the lane marking appearances, for example, dotted lines, solid lines, circular reflectors, white or yellow; (2) variations of the type of road, such as tunnels, bridges, highways and city streets; (3) variations of the time of day, for instance, vehicles in the day time and vehicles at night appear to be totally different; (4) variations of the occlusions, such as pedestrians, bicycles, motorcycles and vehicles that occluding the lane markings; and (5) the presence of shadows of trees and buildings that affects the appearance of lane markings.

Data-driven methods allow for variations of the appearance of the lane markings, and use these variations to increase the heterogeneity of the samples. The more variations contained in the training samples, the better learned of the lane marking model. Lane markings have obvious texture characteristics, so different handcrafted texture features, such as haar-like [4] and local binary pattern [5] could be employed in lane markings detection. Machine Learning algorithms, such as adaboost [6], SVM [7], deep convolutional neural networks [8] were evaluated in literature on lane detection.

At present, the most accurate method is too slow to use (sliding window + CNN [8]), and the fastest method [9] is not accurate enough to use. In this paper, we present a lane marking detection approach that is practical to implement, which is distinguished from the previous ones in the following ways: (1) it is an end-to-end approach to detect small size object, combining multilayer feature maps effectively and efficiently: down-sampling and up-sampling are replaced by convolution layers; (2) adaptive context information is obtained to increase accuracy; (3) it is extended from anchor generation algorithm in Faster RCNN [10], combining

* Corresponding author.
E-mail addresses: tianyan@zjgsu.edu.cn, ty_duck@163.com (X. Wang).

https://doi.org/10.1016/j.neucom.2017.09.098
0925-2312/© 2017 Elsevier B.V. All rights reserved.

sub-pixel sliding window for small objects detection; (4) a new and bigger road image database is built, so the lane markings detection algorithm proposed here could be evaluated better. Our Tian Traffic dataset and the model described in this paper can be downloaded from the author's website.[1]

2. Related work

A milestone in object detection was the Region Convolutional Neural Network (R-CNN) [11]. Ren et al. [10] combined object detection into a unified network, which is called Faster R-CNN. In their work, a Region Proposal Network (RPN) was used to extract candidates. The RPN and the Fast R-CNN [12] both were used to classify the candidates through sharing features based on the Visual Geometry Group 16 (VGG16) model [13].

2.1. Research on small object detection

The RPN approach showed encouraging performance with several hundred candidates in general object detection through using a deep network model, such as the VGG16. However, it still cannot identify and precisely localize small objects, such as lane markings, traffic lights and traffic signs, which is mainly due to the coarseness of its feature maps.

Based on the characteristic of the convolution layer and the pooling layer, it is known that the first layer of the network contains the most location information, while the last contains the least location information. Hariharan and Arbelez [14] proposed hypercolumn feature which up-sampled and combined outputs from all network layers to locate the object and later Hu and Ramanan [15] employed this feature to detect tiny faces. Kong et al. [16] developed a novel hyper feature that deconvolution layer worked on deep layer, and max-pooling worked on shallow layer so that multiple levels of the convolution network obtained feature maps with the same size and could concatenate together, however, it took 1.14 seconds to process each image.

Liu et al. [17] put forward Single Shot MultiBox Detector (SSD) which discretized the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. Although it eliminated proposal generation and subsequent pixel or feature resampling stage and encapsulates all computation in a single network, it still cannot identify and precisely localize small objects.

2.2. Research on context cues

Gidaris and Komodakis [18] proposed multi-region CNN features that original candidate, half boxes, central regions, and border regions were cropped and concatenated to get a robust feature. In their follow-up work [19], considering an initial candidate bounding box, a likelihood value was obtained using the edge cue and context information from inner and outer of the bounding box and then object proposal moved to a new position with a higher likelihood by a regression.

Ghodrati et al. [20] attained object location coarse-to-fine by Inverse Cascade, among which Structured Random Forest was employed in the second layer on the convolution map to estimate the boundary. Then, intersection between the boundary and the candidate box was utilized to get the proposal, depending on the low-level feature. Therefore, the accuracy was not steady. Besides that, it costed more than 0.3 s to process a single frame.

Bell et al. [21] utilized contextual information outside Region-of-interest (ROI) based on spatial recurrent neural networks, and used skip pooling to extracted multi-scale information inside ROI. This method worked well for large size object. However, in terms of small object, it was not robust as the inside information was sensitive.

Zagoruyko et al. [22] added four region crops to their model with "foveal" fields of view of 1 $\times$, 1.5 $\times$, 2 $\times$ and 4 $\times$ of the original proposal box all centered on the object proposal, but there still was no explanation about how to choose scale ratio.

3. Network framework

Our model has 20 convolution layers, and 5 pooling layers with 4 fully-connected layers. This network architecture is a modification of VGG16 including 2 convolutional layers in stage 1(Conv1_1, Conv1_2) and stage 2 (Conv2_1, Conv2_2), 3 convolutional layers in stage 3 (Conv3_1, Conv3_2, Conv3_3), stage 4 (Conv4_1, Conv4_2, Conv4_3), and stage 5 (Conv5_1, Conv5_2, Conv5_3). In the pre-trained VGG16 model, there is a pooling layer at the end of each stage (Pool1, Pool2, Pool2, Pool4, Pool5 in all), and we delete the final pooling layer (Pool5) to increase small object detection capacity and add ROI pooling layer to make this method adaptive to feature map size. Architecture of our network is shown in Fig. 1. Owing to the space limitation, we omit parts that do not play an important role in the framework.

3.1. Feature generator

The first layer of the network contains the most location information, while the last layer contains the most semantic information. As a result, multi-level maps are combined to detect small objects. Our approach was different from [14–16]. In [14–16], different sampling strategies were employed for different layers, for example, max pooling layer on the lower layer is used to carry out sub-sampling task, and un-pooling and deconvolution operation (Deconv) on the higher layers is to conduct up-sampling task. These sampling strategies made it difficult to be learned, and it also took a long time to train the deconvolution layer.

We implemented an end-to-end approach to generate discriminant feature maps. In order to increase semantic information, max pooling layer was replaced by a convolution layer with stride 2 (Conv6_1), and the deconvolution layer was substituted by a sub-pixel convolution layer (Sub-pixel Conv) owing to efficiency reason, and details are deliberated in Section 3.2.

Feature maps from stage 3, 4, and 5 (f3_3, f4_3 and f5_3 in Fig. 1), were concatenated to generate hyper feature maps with 1536 channels. A convolutional layer conv7-1 was added before the ROI pooling layer. Three advantages are introduced due to the structural modification: (1) the channel number of hyper feature maps is significantly reduced (from 1536 to 64); (2) the sliding window classifier is simpler; (3) the kernel size in conv7-1 is modified from 3 $\times$ 3 to 1 $\times$ 1 to restrict the receptive field of the convolution layer.

The output feature maps were fed into two sibling 1 $\times$ 1 convolution layers, respectively, a box-regression layer (conv7-3) and a box-classification layer (conv7-2), on which object location and classification confidence were computed respectively.

3.2. Sub-pixel convolution

Given r to be the upscaling ratio, and the channel numbers for I/O feature maps to be C, the I/O feature maps were represented as real-valued tensors of size $H \times W \times C$ and $rH \times rW \times C$, respectively.

The deconvolution layer proposed in [23] could be multiplication of each input pixel by a filter element-wise with stride r, and sums over the resulting output windows were also known as back-

[1] http://www.yaletian.com/paper/nc17_lmd/ [2017.06.29].

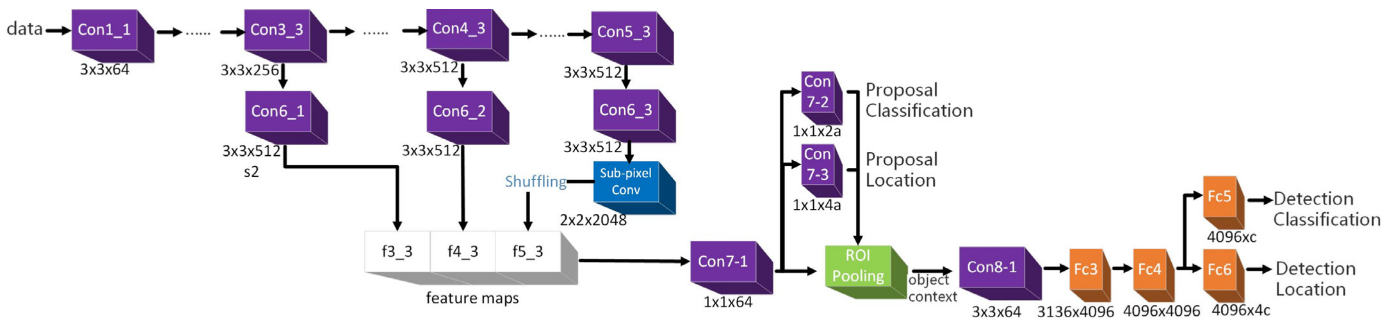


Fig. 1. Architecture of our network. ‘f3_3′, ‘f4_3′ and ‘f5_3′ are feature maps from stage 3, 4, and 5. ‘a’ is number of anchors, and ‘c’ is number of class.

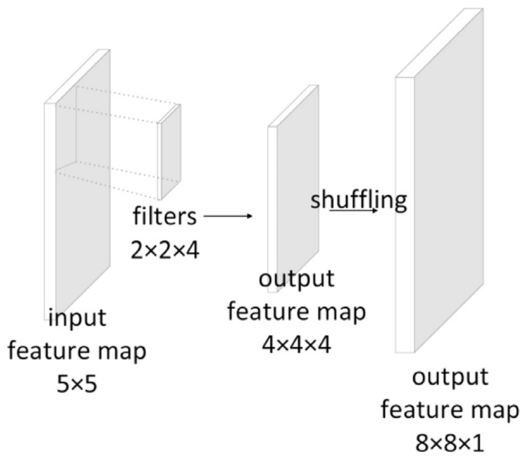


Fig. 2. Illustration of sub-pixel convolution for a single feature map. The grey areas represent padding with zeros.

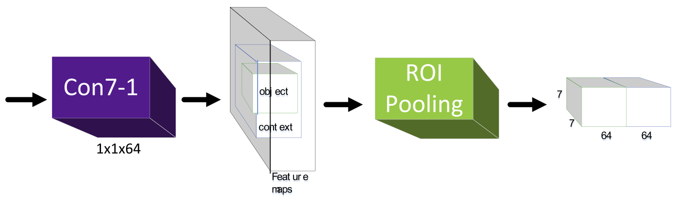


Fig. 3. Illustration of object detection with context information. Features from an object region (green cube) and a context region (blue cube) were stacked together immediately after ROI pooling. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

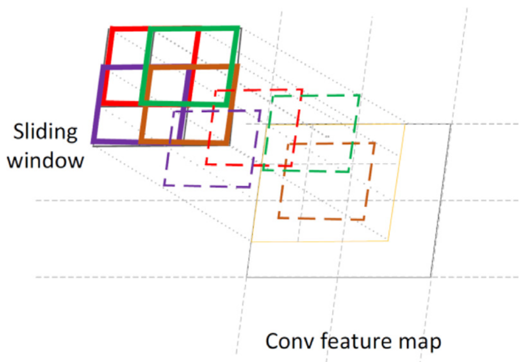


Fig. 4. Illustration of anchors in our network. Bounding box with different color represents specific proposal.

wards convolution. However, any reduction (summing) after convolution is expensive.

Our approach was based on sub-pixel network [24] with 4 upscaling filters for each feature map was shown in Fig. 2. First, the kernel was convolved with the input directly. Then, instead of producing one high resolution feature map, r^2 channels were outputted and periodically shuffled to recreate the final feature map. The procedure was modeled as the following equation:

$$\mathbf{Y} = \mathrm{PS}(\mathrm{W}_L * \mathbf{X} + \mathbf{b}_L), \tag{1}$$

where the size of the convolution operator W_L was $n_{L-1} \times r^2 C \times k_L \times k_L$. PS was a simple periodic shuffling operator that rearranged the elements of a $\mathrm{H} \times \mathrm{W} \times \mathrm{C} \cdot r^2$ tensor to a $r\mathrm{H} \times r\mathrm{W} \times \mathrm{C}$ tensor. In addition to that, periodic shuffling was a bit operation and could be efficiently implemented.

3.3. Context

Context have been proved to be useful for object detection, for example, context information could be modeled by one or multiple regions around the object location in [18,22]. Nevertheless, these context regions were set manually.

In this work, we proposed to utilize context outside proposal because we focused on small object detection. According to Fig. 3, after super feature was obtained with the method introduced in Section 3.1, we used a method similar to hypercolumn feature [14] to obtain a heatmap. Any pixels near the proposal and activated on the heatmap was involved in the context region. Features from an object region (green cube) and a context region (blue cube) were stacked together immediately after ROI pooling. Besides, an extra convolutional layer conv8-1 without padding was used to reduce the number of model parameters, which is conductive to compressing redundant context and object information, without loss of accuracy, and guaranteed that the number of model parameters was approximately the same.

3.4. Anchor generator

This section demonstrates how our neural network generates anchor proposal candidates (see Fig. 4). We show how our model is capable of generating candidates that are much smaller, which makes it a more effective architecture for small-size object identification.

In Faster R-CNN, a window slides over the convolution feature map on the last shared conv layer (conv 5–3 in the VGG16) in order to generate region candidates. Supposing that the conv feature map was $n \times n$, and each position in the feature map could generate a candidate bounding boxes (anchors) with scales s and ratios z. However, when objects in an image are quite small, they do not overlap with any of the anchors, and the system cannot collect enough (positive) samples to train the deep convolutional neural network.

By contrast, our model can detect these small objects like lane marking patches by extending the anchor generator framework, as elaborated in Fig. 4. We restricted the number of anchor ratios since only specific objects (e.g. lane markings) needed to be detected and localized. As the resulting feature map was too coarse to localize the object, we added a sub-pixel transition sliding window in the feature map, so each pixel in convolution feature maps yielded a anchors. Through conducting trial and error experiments, we found that $a = 4$ was effective for the detection task, which increased the possibility that there were enough anchors to interact with the round truth.

3.5. Loss function for learning patch proposals

The VGG16 network has been converted to a multi-task network. In the meantime, the multi-task network can complete numerous relevant tasks simultaneously.

The proposal index in a mini-batch is assumed to be i, and $p_i$is the probability which is predicted by the network. When sample $i$is positive, the ground truth label $c_i$is 1. In contrast, the ground truth label $c_i$is 0 in the case of negative sample i. $\mathbf{P}_i = (P_x, P_y, P_w, P_h)$ means the scale parameter and position parameter of the candidate proposal bounding box. $(P_x,\ P_y)$ is the center point of the bounding box. $(P_w,\ P_h)$ is the width and height of the bounding box, and $\mathbf{G}_i = (G_x, G_y, G_w, G_h)$ is the ground truth. L_{cls} is supposed to be the (binary) classification error. L_{reg} is the regression error of the predicted position. The function of the overall loss can be expressed as below:

$$L(p_i, c_i, \mathbf{P}_i, \mathbf{G}_i) = \frac{\alpha}{N_{cls}} \sum_i L_{cls}(p_i, c_i) + \frac{1}{N_{reg}} \sum_i L_{reg}(\mathbf{P}_i, \mathbf{G}_i), \tag{2}$$

where N_{cls} and N_{reg} are normalization coefficients, and αbalances the two kinds of loss.

The classification error $L_{cls}(p_i,\ c_i)$ represents log loss over the two classes (foreground vs. background)

$$L_{cls}(p_i, c_i) = -\sum_i c_i \log(p_i). \tag{3}$$

The candidate bounding box is supposed to be represented as $\mathbf{P}_i = (P_x, P_y, P_w, P_h)$, and $\Phi_5(\mathbf{P}_i)$represents a corresponding feature in the convolution map. Similarly, the corresponding ground truth can be expressed as $\mathbf{G}_i = (G_x, G_y, G_w, G_h)$. Two kinds of mapping for f and $t$should be found so that$f(\mathbf{P}_i) \approx t(\mathbf{G}_i)$.

If the candidate is close to the ground truth, f and t can be expressed as a linear regression model. It is assumed that the mapping of one vector into the other vector of the proposal is $f(\mathbf{P}_i) = [\mathrm{f}_x(\mathbf{P}_i), f_y(\mathbf{P}_i), f_w(\mathbf{P}_i), f_h(\mathbf{P}_i)]$ with scale transformation $(f_w(\mathbf{P}_i),\ f_h(\mathbf{P}_i))$and transition transformation $(\mathrm{f}_x(\mathbf{P}_i),\ f_y(\mathbf{P}_i))$. If $\mathbf{W}_*$ represents the learned parameters in f, then $f_*(\mathbf{P}_i) = \mathbf{W}_*^T \Phi_5(\mathbf{P}_i)$. These two mappings ($f$ and t) are not learned simultaneously, and the real transformation $t$can be divided into scale transformations $(t_w,\ t_h)$ and transition transformations $(t_x,\ t_y)$, which can be represented as

$$t_x = (G_x - P_x)/P_w \tag{4}$$

$$t_y = (G_y - P_y)/P_h \tag{5}$$

$$t_w = \log(G_w/P_w) \tag{6}$$

$$t_h = \log(G_h/P_h). \tag{7}$$

Then, the position of the candidate object can be fine-tuned by two convolution layers. The optimized loss function is

$$L_{reg}(\mathbf{P}_i, \mathbf{G}_i) = \sum_{i=1}^{N} \|\mathbf{P}_i - \mathbf{G}_i\|^2 + \eta \sum_{i=1}^{N} \left(t_i - \mathbf{W}_*^T \Phi_5(\mathbf{P}_i)\right)^2 + \lambda \|\mathbf{W}_*\|^2, \tag{8}$$

where $\mathbf{t}_i = (t_x, t_y, t_w, t_h)$ is the vector format, and N is the sample number in a mini-batch. The first term is the localization error. The second term is the regression error. The last term is the regularization term, and λ and η are influencing factors selected by parameter adjustment. The method of finding the optimal influence factors is unknown. Therefore, the engineering experience is used to adjust these parameters based on validation and training data. A specific influencing factor is changed while other parameters are fixed to learn the model by the training data in accordance with the loss in Eq. (8). The model is evaluated on the validation data based on the increase of the average precision. In the case that average precision increases, the value of the new influencing factor is fixed, and vice versa. Afterward, the other influencing factor is changed, and the same method is adopted to evaluate whether the value of the new factor has been accepted. Although this method is greedy, it works well in the system and use $\eta = 0.1$, $\lambda = 0.05$ in our experiments. $\mathbf{W}_*$ is obtained by gradient descent method.

4. Lane marking detection and the road equation

This section explains how the lane marking patches are detected and located by an end-to-end deep convolutional neural network. Compared with other approaches by manually-designed features, the deep convolutional neural network can extract more discriminative features. Comparing with the approaches on the basis of a sliding window, an end-to-end deep convolutional neural network can detect lane markings in a real-time manner by GPU parallel computing. In addition, the method of detecting lane marking patches can precisely obtain the direction and the position of the lane markings, which cannot be realized by other approaches. A lane equation is employed to warn lane departure. What's more, a lane equation is determined by the large-scale dataset optimization method [25].

4.1. Learning

Two models are trained in the KITTI-ROAD dataset and the Tian Traffic data. In Section 5, datasets are presented to explain the annotation process. Tian Traffic model is trained on 16,000 images, and KITTI-ROAD model is trained on 289 images.

Sample instances that can hardly be identified are required. In other words, whether they contained lane markings for a road is unclear. Obviously, most in-lane videos can be recorded by cars driving in-lane. Actually, both of the two datasets contained a great number of easy examples and a small number of difficult examples. Therefore, a method on base of online hard example mining (OHEM) [28] is used, so that the training process is more efficient and effective. In each SGD iteration, a mini-batch was obtained as below: Firstly, a total of N images were sampled from the training dataset, and candidates for objects with intersection over union (IoU) overlapping with a ground-truth bounding box of at least 0.5 were selected as positive samples, and other candidates with IoU overlapping with a ground-truth bounding box of at most 0.2 were selected as negative samples. Secondly, a read-only network computed the loss value and performed a forward propagation (sum of regression loss and classification loss) for all of the input proposals. Hard examples were selected by sorting the candidates by loss and taking the R/*N* examples for which the read-only network performs worst. In practice, $N = 2$ and $R = 128$. Overlapping candidates are projected to the same region in the conv feature map. Therefore, deduplication was performed through non-maximum suppression (NMS) by iteratively selecting the candidate of which the loss is the highest, and all candidates with lower loss that overlap with the selected region were removed. Finally, regular network model computed forward and backward propagations only for difficult examples, accumulated and passed the gradients to the neural network model.

In our model, the proposals are obtained by the method in Section 3.4. Conv 1–1 layer to conv 5–3 layer were obtained by the pre-trained VGG16 model. New layers conv 6–1 to conv 8–1 were initialized by a fast and simple data-dependent initialization procedure [27] in which the initial weights of a network are given. Therefore, all units in the network are roughly trained at the same rate to avoid vanishing or explode gradients.

4.2. *Inference for lane marking*

The inference procedure tests the image in the way that the network executes a forward propagation, and outputs the location of each lane marking patch and the classification confidence. The results indicate that many objects not to be positive examples are detected. Consequently, the performance is improved by filtering images (such as those with just sky or vegetation), so that the false alarms can be removed and the precision is increased.

The images have been filtered in multiple ways. Regions-of-interest can be designed in accordance with the temporal information in video sequence, such as by images from the previous frame with lane markings. Alternatively, texture, color or edge information of the lanes or lane markings are used to obtain a RoIs. However, it is found that RoIs could be better determined by geometric properties, such as the features of perspective: parallel lines pass through the same point on the horizon– the vanishing point. Afterwards, the horizon location can be exactly expressed by the intersection of lines at the vanishing point.

4.3. *Inference for road equation*

Each of the patch center is recorded for the obtained lane marking patches. Most of the road markings are straight, except around corners. Therefore, the lane equation can be represented as a linear function

$$y = a_0 + a_1 x, \tag{9}$$

where (x, y) is the center of the patch. Linear function coefficients can be solved by optimization

$$\min_{(a_0, a_1)} \sum_i [y_i - (a_0 + a_1 x_i)]^2 + \lambda_0 (a_0 - a_0^*)^2 + \lambda_1 (a_1 - a_1^*)^2, \tag{10}$$

where the first term represents the error. The second term is regularized, and it represents the temporal smoothing. a_0^* and a_1^* are the corresponding parameters of the lane function getting from the previous frame calculation. λ_0 and $\lambda_1$can balance different components. Our solution is based on a large scale dataset optimization method [25].

5. Data sets

We used two data sets for experiments. One set has been extracted from the KITTI image data for road detection, and the Tian Traffic data we created ourselves.

The KITTI ROAD data set has been extracted from a computer vision data set known as KITTI.[2] The KITTI-ROAD data set consists of about 600 frames (375 × 1242 pixels), which are separated into 289 training and 290 testing images. The recordings were taken on five different days, all of which, had relatively low traffic density. Therefore, the road was often completely visible.

Our Tian Traffic data had a total of 16,000 training images and 3000 validation and test images. We created the images by using a video camera fixed on the front window of each of four vehicles (Ford Mondeo, Volkswagen Tiguan, Volvo S60, and Citroen C4L). All the cameras had vertical height 1.2 m−1.6 m, and their optical directions were almost horizontal forward. Locations of collection included urban, highway, tunnel, and bridge, which can be found in Fig. 5. The cars travelled during day, night and at sunset. The extracted data included daytime segments, nighttime segments, and sunset segments, some with poor resolution. To make 35,600 frames in total, each of these segments lasted 3 minutes (5400 frames). Training data and testing data were from different video segments, and different videos were recorded on different roads, therefore, there were no lane markings in both training data and testing data. Basically, lane marking segments were divided automatically by fixed vertical pixel interval in the image to represent positive samples, and segments anywhere in the photo were randomly selected to represent negative samples. Therefore, lane marking segments overlapped with each other in some cases. In some situations, if the annotator missed some lane markings before the automatic division process, he would annotate the lane marking patches one after one by hand. The manual annotation examples could be seen in the lower right corner of Fig. 5.

We developed our own web-based platform to distribute the images to ten anonymous annotators. As shown in Fig. 5, the data included both actual lane markings as positive data and non-lane markings as negative data. We asked annotators to mark each of 4 corner points of an actual lane segment and to skip over those that were not road lanes. Each image was annotated once in order to complete the entire set of 35.6 K images.

As for the results of this annotation, a total of 23,000 images in the whole set of 35,000 images have lane markings, and 19,000 images among which are correctly annotated. Some samples can be observed in Fig. 5, where the negative samples are in green bounding boxes, and the positive samples are in red bounding boxes. Ten individuals annotated the images in two months.

The random or fixed ratio of width to height of the lane marking patches can be set. In our experiment, the width/height ratio is fixed to 2.0 so as to simplify the comparison between the adaboost method and the method proposed in our paper.

6. Experimental results

6.1. *Parameters of the experiment for lane marking detection*

6.1.1. *Hardware and software environment*

We carried out the experiments on a workstation with an Intel i7-4790 3.6 GHz CPU, 32GB memory, and NVIDIA GTX Titan X graphics. The algorithm is on the basis of the Caffe library [26] so that the computational efficiency and the performance can be verified.

6.1.2. *Models for training and testing*

There are two models, and KITTI-ROAD data are used because it has been publically available for several years. However, the KITTI-ROAD model was trained by a small data set (289 images for training). The main experiments in this paper were carried out and evaluated in our data set (16,000 images for training), and the evaluation experiments are implemented by the Tian Traffic model with 3000 images to be verified and tested.

[2] http://www.cvlibs.net/datasets/kitti/eval_road.php.

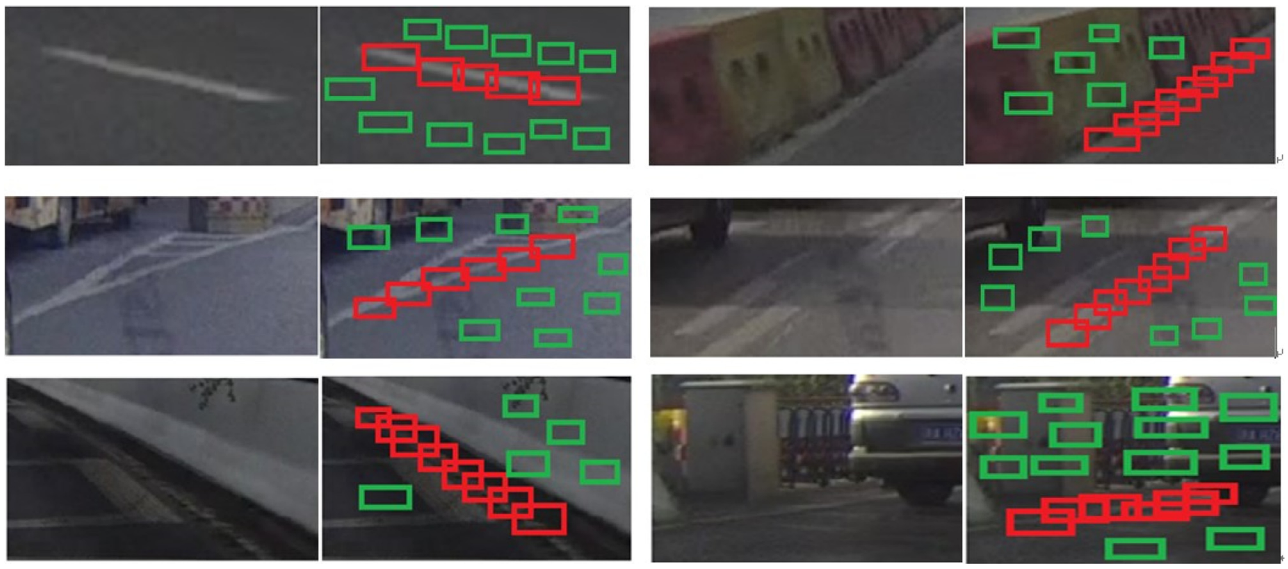


Fig. 5. Tian Traffic data examples with variations in appearance. The raw image in each pair is on the left, and the image with positive markings (red boxes) and negative markings (green boxes) is on the right. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

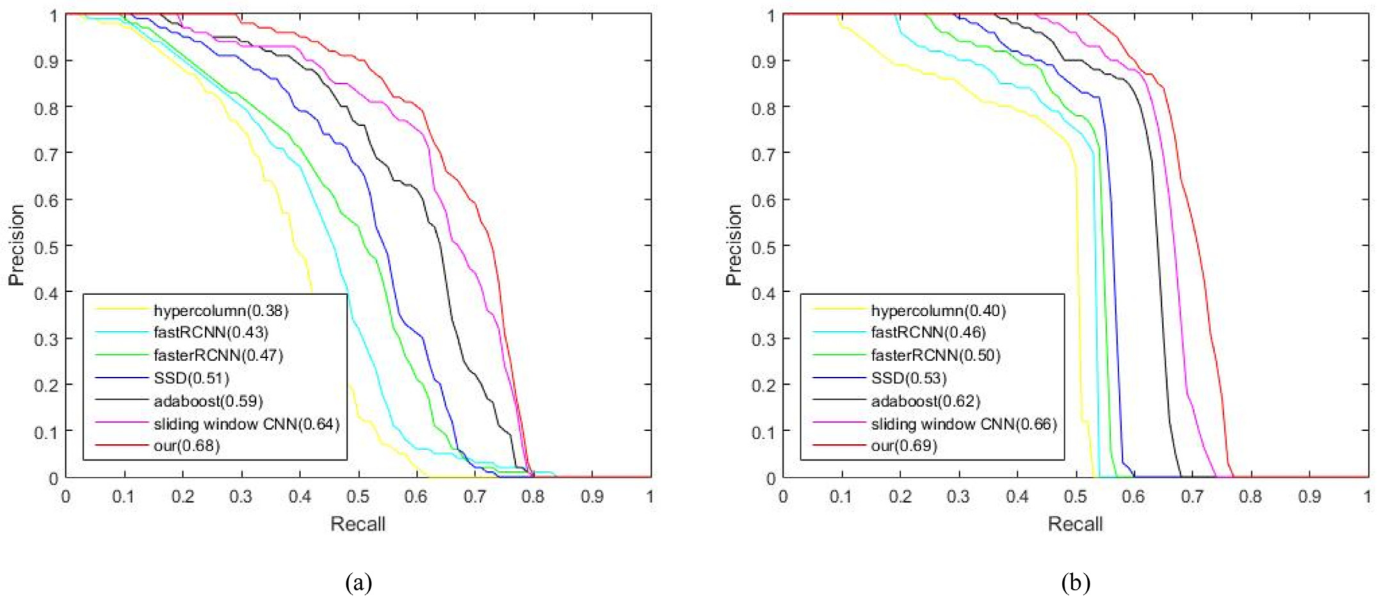


Fig. 6. (a) Precision-recall curves on the KITTI-ROAD dataset. (b) Precision-recall curves on the Tian Traffic dataset. APs were reported in brackets.

6.1.3. Evaluation criteria

The evaluation criteria of lane marking localization are Average Precision (AP). Precision is the proportion of all of the examples detected that are positive ground truth, and recall is the proportion of positive ground truth images detected in the sample. The precision/recall curve is computed according to the ranked output.

6.2. Result output for the lane marking detection experiment: KITTI road model

Fig. 6(a) showed the precision-recall curves and APs of lane marking patches detection on KITTI-ROAD dataset (Urban Marked Multi-lane Road). Through using the source code from the authors, we compared our method with other state-of-the-art detection approaches. Detection confidence threshold was tuned to vary recall and precision values and obtain precision-recall curves.

Due to the limitation of the feature map size, the accuracy of the methods based on proposal, such as Fast R-CNN, Hypercolumn, and Faster R-CNN is limited. However, sliding window approaches, such as sliding window CNN and Adaboost, can get better results compared with proposal methods, because each location and scale of the candidate proposal can be verified. The AP of our algorithm is 68%, which is 4% higher than that of sliding window CNN.

Fig. 7 shows a sample of the result output. The Adaboost approach is employed to obtain the blue bounding boxes [2], and Faster R-CNN is used to acquire the green bounding boxes [10]. Our algorithm gets the red bounding boxes. As shown in the figure, despite of the complex background of a traffic scene, such as roads, buildings and trees, it is easy to detect the lane markings with a strong pattern. What's more, the adaboost approach can verify all positions in the region of interest, namely, $200 \times 1200 = 240,000$ candidates, while only 300 proposals are generated by Faster R-

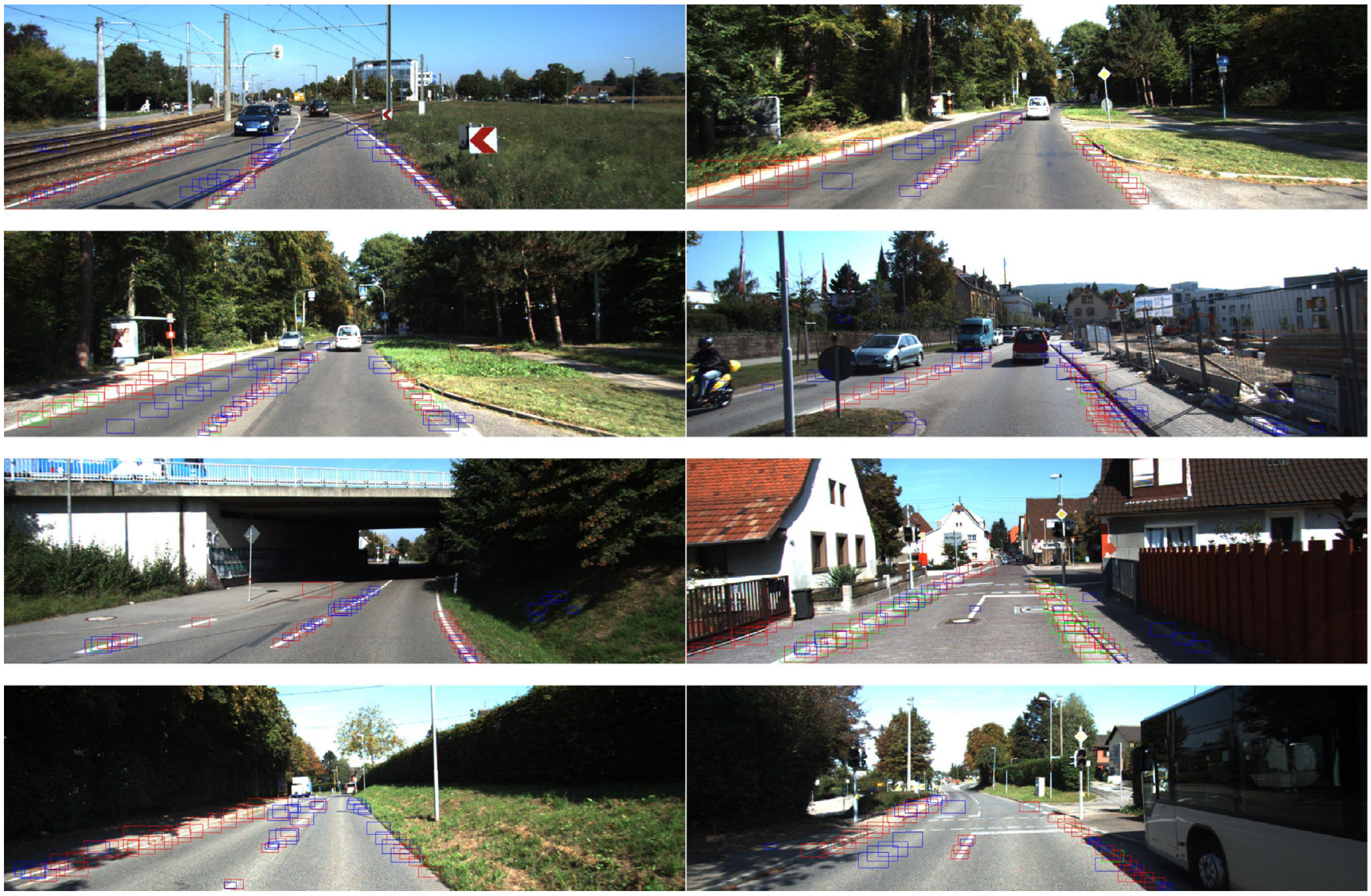


Fig. 7. KITTI-ROAD model: Lane markings detection with proposal anchors in KITTI-ROAD dataset. Blue are Adaboost, Green are Faster R-CNN and Red are our approach. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

CNN. As a result, more candidates can be evaluated by Adaboost compared with the Faster R-CNN, and the theoretical recall ratio is better.

To our surprise, the recall was greatly improved by just adjusting the receptive field of the neural network, and through adding some position variation to the feature map, providing as many as 4 times more proposals in our experiment. Adaboost extracted hand-crafted low-level features and balanced all the samples in the training data to find some statistical parameters to reasonably describe the object's pattern. However, Adaboost did not use high-level features, which showed that weaker non-linear classification capabilities than our deep convolution neural network. That is why Adaboost obtained more false alarms than Faster R-CNN and our algorithm. Although the number of false alarms might be reduced by adding more layers in a cascade approach, the number of the weakly classified examples would increase greatly, and there is no guarantee that adaboost would increase in overall performance.

Fig. 7 shows that some of the lane markings were missed because the feature map was still relatively coarse. In our approach, the images were resized to their original resolution, and 5 pooling layers were employed in the forward propagation process. Thus, the base anchor size was 16 $\times$ 16, and even with a sub-pixel sliding window on the feature map output, each proposal interval was approximately 8 square pixels.

6.3. Result output and performance for the lane marking detection Experiment: tian traffic data

Detection results of Adaboost (in blue), Faster R-CNN (in green), and our approach (in red) are shown in Fig. 8. Adaboost gave dense detection bounding boxes, but it could not handle some situations in which the image had different directions or scales. That is why some lane marking segments in Adaboost were not detected. Faster R-CNN had a larger receptive field, therefore, even with small size candidate proposals, the output classification confidence was not stable for the reason that too much context information was included in the input. Our network had a small size receptive field, hence it could detect small size lane marking patches. However, the detection result was relatively sparse owing to the stride of the sliding window on the feature map.

Our Tian Traffic data had a total of 16,000 training images and 3000 testing images. We trained our deep convolution neural network detection model through using 16,000 training images. Table 1 and Fig. 6(b) were created using our entire 3000 images testing set. To obtain the precision-recall corresponding values, detection confidence threshold was tuned from 0.01 to 0.99 (with a stride 0.01), and then the AP was acquired by the precision-recall corresponding values. In addition, to compare our approach with other approaches, we also reported recall and precision values in fixed detection confidence threshold, and this detection confidence threshold was chosen according to the rule that the sum of precision-recall corresponding values was maximized. In the present experiment, the detection confidence threshold is 0.60.

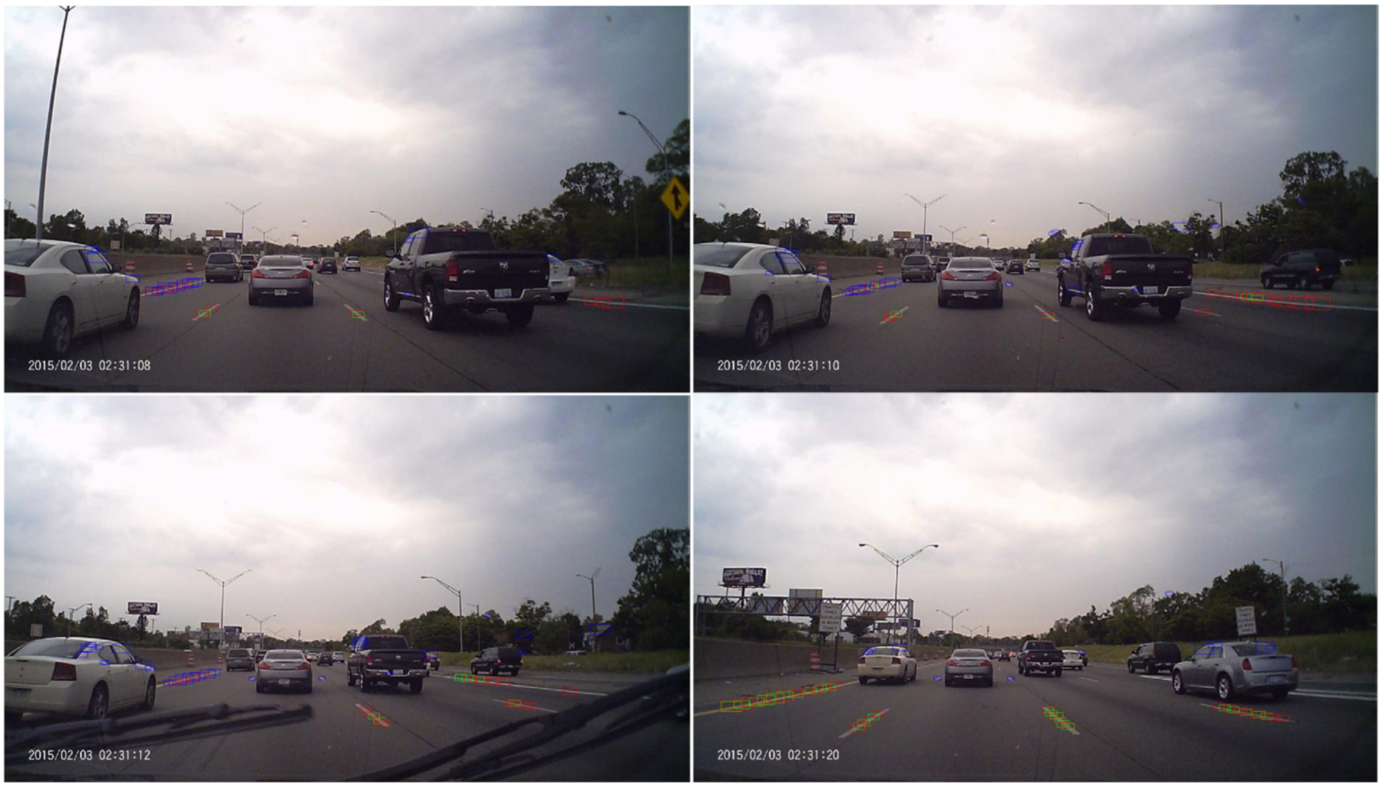


Fig. 8. Tian Traffic model: Tian Traffic lane markings dataset with proposal anchors from Adaboost (blue), faster R-CNN (green), and our approach (red). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 1
Detection result comparison on our lane marking dataset.

Algorithm	Performance (AP)	Recall (%)	Precision (%)	Speed (milliseconds/frame)
Hypercolumn [29]	40.4	52.8	68.4	214
Fast R-CNN [12]	45.8	56.3	71.8	2213
Faster R-CNN [10]	49.5	57.6	73.2	130
SSD [17]	52.8	58.5	75.9	71
Adaboost [6]	61.7	64.1	68.7	**56**
Sliding window + CNN [8]	66.1	64.5	81.1	75,000
Our approach	**69.2**	66.4	83.5	206

Sliding window approaches like adaboost and sliding window CNN obtained better results than Faster R-CNN because they verified each location and scale of the candidate proposal. Adaboost was extraordinarily fast, processing a single frame in only 56 ms. Techniques could be employed to speed the detection process further. Lane markings detection by Adaboost was fast because the pattern of lane markings was simple and clear. For example, there were only 2 weak classifiers in the first layer and 6 weak classifiers in the second layer. However, Adaboost could not converge easily when it came to more than 20 layers, hence, Adaboost could currently be employed for lane markings detection with limited accuracy. Sliding window CNN gave the second best result in the comparison, however, each proposal was resized and sent into the network and thousands of forward propagations were calculated, which had contributed to greatly increasing computational complexity.

Given an image of 1280×1920pixels, the Faster R-CNN resized the image to 600×1000 at first, then the convolutional network extracted features (this took 65 ms), and finally false alarms were filtered (another 35 ms). Besides, the total Faster R-CNN processing time was about 130 ms. Although our method was not as fast as Adaboost now, GPU computing developed fast, and our method would speed up when more CUDA units were deployed.

Our approach obtained the best result through building hyper feature maps combing multi-level information, adding context cues for detection, modifying the receptive field of the neural network, and adding some location variations to the image (that is, the feature map). Rectangular-box anchors in Faster R-CNN were varied with three scales and three ratios, which is the default setting in Faster R-CNN, while candidate anchors were varied only in location. Through adding 8 pixels each time, we changed the position of bounding box in the image. Therefore, we could be sure of getting more positive anchor/bounding boxes.

6.4. Lane equation

We calculated a lane equation by fitting detected lane marking patches together. We used a linear function described in Eq. (9) for real-time processing and the RANSAC approach [29] was used to filter outliers. We used a threshold of 3 bounding boxes to distinguish regular data from outliers, we determined threshold by experimentation. The Lane Equation results are shown in Fig. 9. Geometry information could be employed to further improve the result. For example, we could use the vanishing point to obtain the end of the line equation since all the lane markings intersect at the vanishing point.

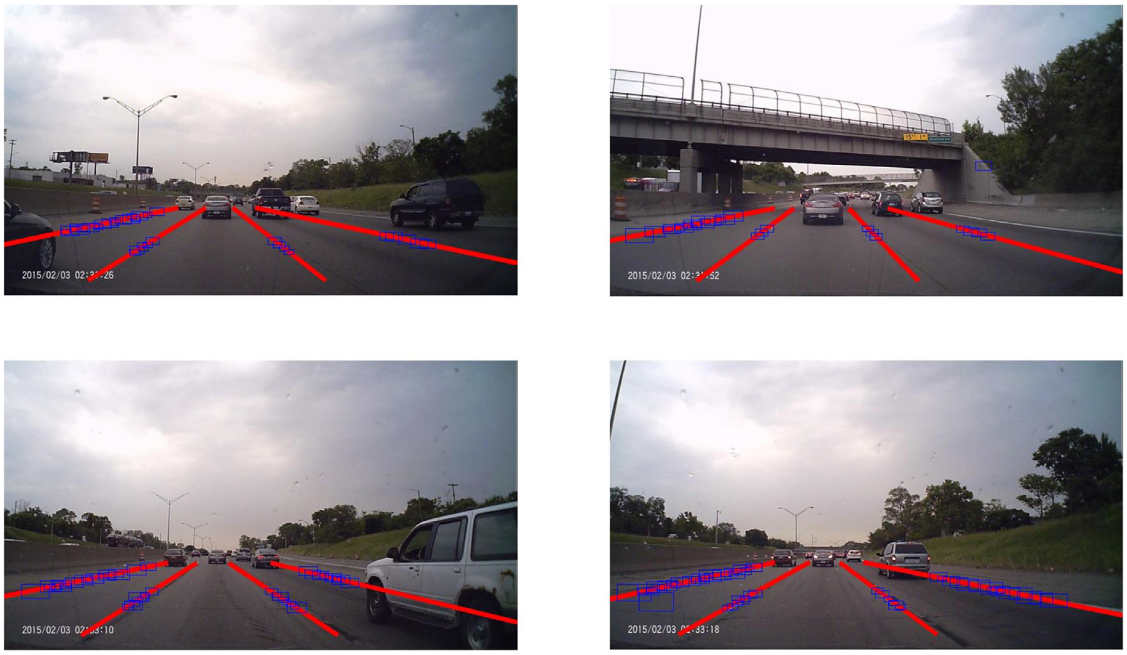


Fig. 9. Lane function created by fitting detected lane marking patches. Lane equation are lines in red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

7. Conclusion

To conclude, we have proposed a method that identifies and locates small objects in images to some confidence value based on a multi-task network. Our method has proved to be fast while it obtains better accuracy than the most accurate method. Continuing research would use the geometry of the object to improve detection of that object.

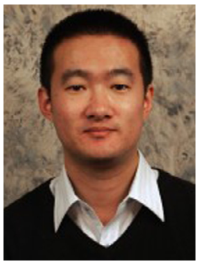
Acknowledgment

This work was supported in part by National Natural Science Foundation of China (No. 61472363, U1609215, 61602407, and 61672460), Opening Foundation of State Key Laboratory of Virtual Reality Technology and System of Beihang University (No. BUAA-VR-16KF-19).

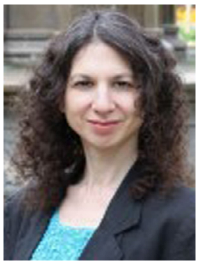
References

[1] M. Bertozzi, A. Bmggi, GOLD: a parallel real-time stereo vision system for generic obstacle and lane detection, IEEE Trans. Image Process. 7 (1) (1998) 62–81.
[2] M. Beauvais, C. Kreucher, Building world models for mobile platforms using heterogeneous sensors fusion and temporal analysis, in: Proceedings of the ITS, 1997, pp. 230–235.
[3] B. Yu, AK. Jain, Lane boundary detection using a multiresolution hough transform, in: Proceedings of the IEEE International Conference on Image Processing, 1997, pp. 748–751.
[4] S. Han, Y. Han, H. Hahn, Vehicle detection method using Haar-like feature on real time system, World Academy of Science, Eng. Technol. 59 (2009) 455–459.
[5] R. Gopalan, T. Hong, M. Shneier, R. Chellappa, A learning approach towards detection and tracking of lane markings, IEEE Trans. Intell. Transp. Syst. 13 (3) (2012) 1088–1098.
[6] M. Everingham, S.M.A. Eslami, L. Van Gool, C.K.I. Williams, J. Winn, A. Zisserman, The pascal visual object classes challenge: a retrospective, Int. J. Comput. Vis. 111 (1) (2015) 98–136.
[7] Z.W. Kim, Robust lane detection and tracking in challenging scenarios, IEEE Trans. Intell. Transp. Syst. 9 (1) (2008) 16–26.
[8] B. Huval, T. Wang, S. Tandon, J. Kiske, W. Song, J. Pazhayampallil, M. Andriluka, P. Rajpurkar, T. Migimatsu, R. Cheng-Yue, F. Mujica, A. Coates, A.Y. Ng, An empirical evaluation of deep learning on highway driving. http://arxiv.org/abs/1504.01716, 2015.
[9] J. Redmon, S. Divvala, R. Girshick, et al., You only look once: Unified, real-time object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 779–788.
[10] S.Q. Ren, K.M. He, R. Girshick, J. Sun, Faster R-CNN: Towards real-time object detection with region proposal networks, Adv. Neural Inf. Process. Syst. (2015) 1–9.
[11] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 580–587.
[12] R. Girshick, R-CNN Fast, in: Proceedings of the IEEE Conference on Computer Vision Pattern Recognition,, 2015, pp. 1440–1448.
[13] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: Proceedings of the International Conference on Learning Representations, 2015, pp. 585–592.
[14] B. Hariharan, P. Arbelez, Hypercolumns for object segmentation and fine-grained localization, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2015, pp. 447–456.
[15] P Hu, D Ramanan, Finding tiny faces. https://arxiv.org/pdf/1612.04402, 2016.
[16] Y. Xiang, W. Choi, Y. Lin, S. Savarese, Subcategory-aware convolutional neural networks for object proposals and detection. http://arxiv.org/abs/1604.04693, 2016.
[17] W. Liu, D. Anguelov, D. Erhan, et al., SSD: Single Shot MultiBox Detector, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 21–37.
[18] S. Gidaris, N. Komodakis, Object detection via a multi-region and semantic segmentation-aware CNN model, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1134–1142.
[19] S. Gidaris, Komodakis N, LocNet: improving localization accuracy for object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 789–798.
[20] A. Ghodrati, A. Diba, M. Pedersoli, et al., DeepProposal: hunting objects by cascading deep convolutional layers, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 2578–2586.
[21] S. Bell, CL. Zitnick, K. Bala, et al., Inside-outside net: detecting objects in context with skip pooling and recurrent neural networks, in: Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition, 2016, pp. 2874–2883.
[22] S. Zagoruyko, A. Lerer, T.Y. Lin, et al., A MultiPath Network for Object Detection, BMVC (2016) 74–81.
[23] M.D. Zeiler, G.W. Taylor, R. Fergus, Adaptive deconvolutional networks for mid and high level feature learning, in: Proceedings of the IEEE International Conference on Computer Vision, 2011, pp. 2018–2025.
[24] W. Shi, J. Caballero, F. Huszár, et al., Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1874–1883.
[25] L. Bottou, Large-scale machine learning with stochastic gradient descent, in: Proceedings of the COMPSTAT, 2010, pp. 177–186.
[26] D.J. Ea, Y. Jia, E. Shelhamer, Caffe: Convolutional architecture for fast feature embedding, in: Proceedings of the ACM Multimedia, 2014, pp. 675–678.
[27] P. Krähenbühl, C. Doersch, J. Donahue, et al., Data-dependent initializations of convolutional neural networks, in: Proceedings of the IEEE International Conference on Learning Theory, 2016, pp. 214–221.

[28] A. Shrivastava, A. Gupta, R. Girshick, Training region-based object detectors with online hard example mining, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 761–769.
[29] R. Schnabel, R. Wahl, R. Klein, Efficient RANSAC for point-cloud shape detection, Comput. Gr. Forum 26 (2) (2007) 214–226.

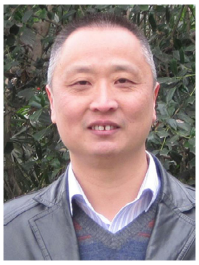


Yan Tian was born in Mudanjiang, China. He received B.Sc. in Communication Engineering from Hangzhou Dianzi University, Hangzhou, China, and Ph.D. degrees from Beijing University of Posts and Telecommunications, Beijing, China, in 2005 and 2011, respectively. Then he conducted as a postdoctoral research fellow (2012–2015) in Department of Information and Electronic Engineering, Zhejiang University, Hangzhou, China. He is currently a Lecture of Computer Science and Technology in the school of Computer Science and Information Engineering, Zhejiang Gongshang University, China. His current interests are machine learning and pattern recognition, and he also works on image and video Analysis.

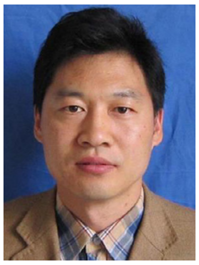


Judith Gelernter received her Ph.D. in information science is from Rutgers University. She is a Research Scientist in the Information Technology Laboratory at the National Institute of Standards and Technology (NIST), where her work is focused on indoor mapping, and also on machine learning in the service of language processing. Prior to her position at NIST, she did research on machine learning mostly for geo-parsing from 2008–2015 in the Language Technologies Institute of the School of Computer Science of Carnegie Mellon University.

Xun Wang received his BSc in mechanics, M.Sc. and Ph.D. degrees in computer science, all from Zhejiang University, Hangzhou, China, in 1990, 1999 and 2006, respectively. He is currently a Professor in the School of Computer Science and Information Engineering, Zhejiang Gongshang University, China. His current research interests include mobile graphics computing, image/video processing, pattern recognition and intelligent information processing. In recent years, He has published over 80 papers in high-quality journals and conferences. He holds 9 authorized invention patents and 5 provincial and ministerial level scientific and technological progress awards. He is a member of the IEEE and ACM, and a senior member of CCF.

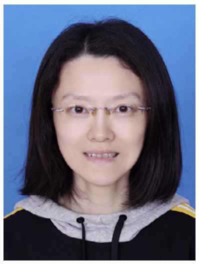


Weigang Chen received the M.S. degree from Zhejiang Sci-Tech University, Hangzhou, China, in 1995, and Ph.D. degree from the department of Computer Science and Technology, Shanghai Jiaotong University, Shanghai, China, in 2004. Since 2004, he is an associate professor with the School of Computer Science and Information Engineering, Zhejiang Gongshang University, Hangzhou, China. His current interests include video and image processing, pedestrian detection and counting, video compression and communication.



Junxiang Gao was born in Tangshan, Hebei province of China. He received BSc in Communication and Navigation from Shanghai Maritime University in 1996, and master's degree and Ph.D. degree from Jiaotong University and Beijing University of Posts and Telecommunications in 2005 and 2010, respectively. Then he employed by Huazhong Agricultural University, China. He is currently the dean of computer science and technology department, an associate professor of Informatics School of the above university. His research interests are data analysis, data mining and application development in the field of agriculture and life science.

Yujie Zhang was born in Shaoxing, China. Now he conducts as a research assistant in school of Computer Science and Information Engineering, Zhejiang Gongshang University, China. His current interests are machine learning and pattern recognition, and he also works on image and video Analysis.



Xiaolan Li was born in Yantai, China. She received the M.S. degree in power engineering from Shandong University, Jinan, China and the Ph.D. degree in electrical engineering from Zhejiang University, Hangzhou, China, in 1999 and 2003, respectively. Then she conducted as a postdoctoral research fellow (2003–2005) in clinical medicine from pharmaceutical informatics institute, Zhejiang University, Hangzhou, China. She is currently a professor of Computer Science and Technology in the school of Computer Science and Information Engineering, Zhejiang Gongshang University, China. Her current interests are biomedical signal processing and pattern recognition. She also works on image and video processing.