Simultaneous Traffic Sign Detection and Boundary Estimation Using Convolutional Neural Network

Hee Seok Lee and Kang Kim

***Abstract*— We propose a novel traffic sign detection system that simultaneously estimates the location and precise boundary of traffic signs using convolutional neural network (CNN). Estimating the precise boundary of traffic signs is important in navigation systems for intelligent vehicles where traffic signs can be used as 3-D landmarks for road environment. Previous traffic sign detection systems, including recent methods based on CNN, only provide bounding boxes of traffic signs as output, and thus requires additional processes such as contour estimation or image segmentation to obtain the precise boundary of signs. In this paper, the boundary estimation of traffic sign is formulated as 2-D pose and shape class prediction problem, and this is effectively solved by a single CNN. With the predicted 2-D pose and the shape class of a target traffic sign in the input, we estimate the actual boundary of the target sign by projecting the boundary of a corresponding template sign image into the input image plane. By formulating the boundary estimation problem as a CNN-based pose and shape prediction task, our method is end-to-end trainable, and more robust to occlusion and small targets than other boundary estimation methods that rely on contour estimation or image segmentation. With our architectural optimization of the CNN-based traffic sign detection network, the proposed method shows a detection frame rate higher than seven frames/second while providing highly accurate and robust traffic sign detection and boundary estimation results on a low-power mobile platform.**

***Index Terms*— Traffic sign detection, traffic sign boundary estimation, convolutional neural network.**

I. Introduction

TRAFFIC sign detection has been a traditional problem for intelligent vehicles, especially as a preceding step for traffic sign recognition which provides useful information such as directions and alerts for autonomous driving or driver assistance systems. Recently, traffic sign detection has received another attention from navigation systems for intelligent vehicles, where traffic signs can be used as distinct landmarks for mapping and localization. Different from natural landmarks such as corner points or edges which have arbitrary appearance, traffic signs have standard appearances such as shapes, colors, and patterns defined by strict regulations. The standard appearances of traffic signs make it efficient and robust to detect and match traffic signs under various conditions, and this forms a primary reason that traffic signs are a preferable choice as landmarks for road map reconstruction.

To reconstruct detected traffic signs to a 3D map, one should have point-wise correspondences of boundary corners of the signs across multiple frames, and then compute 3D coordinates of the boundary corners by triangulation using the camera pose and internal parameters of the camera (see [1] for details of the 3D mapping procedure). For accurate triangulation of 3D position, it is required to estimate the boundary of signs with pixel-level accuracy. However, previous traffic sign detection systems do not meet this requirement as they only estimate bounding boxes of traffic signs. Pixel-wise prediction methods such as semantic image segmentation, which have been applied successfully for road scenes [2]–[4], can be an alternative for boundary estimation. However, it requires time-consuming algorithms that can severely harm the performance of real-time systems for vehicles.

To avoid this inefficiency, we propose a traffic sign detection system where the position and precise boundary of traffic signs are predicted simultaneously using a single convolutional neural network (CNN). Our novel object detection network is based on the recent advances in CNN based object detection for object bounding box prediction [5]–[7], but tailored to predict 2D poses and shape labels of planar targets. The 2D pose of planar targets can be encoded as 8-dimensional vectors, *e.g.,* coordinates of four vertices, and it can be accurately predicted by CNN which predicts the scores of each shape label simultaneously. Using the predicted 2D poses and shape labels, the boundary corners of a traffic sign are computed by projecting the boundary corners of a corresponding template image of the sign into the image coordinate using the predicted pose, as illustrated in Figure 1.

By using the templates of traffic signs, our method effectively utilizes strong prior information of target shapes. This enables robust boundary estimation for traffic signs that are occluded or blurry, which is often challenging in pixel-wise prediction such as contour estimation and segmentation. As a result, our method achieves detection rates higher than 0.88 mean average precision (mAP), and boundary estimation error less than 3 pixels with respect to input resolution of 1280 × 720 pixels. Since projecting boundary corners (matrix-vector products) requires negligible computation time, most of the required computation is from the forward

Manuscript received February 28, 2017; revised July 27, 2017 and November 27, 2017; accepted January 22, 2018. Date of publication March 8, 2018; date of current version May 2, 2018. The Associate Editor for this paper was D. Fernandez-Llorca. *(Corresponding author: Hee Seok Lee.)*

The authors are with Qualcomm Technologies Inc., Seoul 06060, South Korea (e-mail: heeseokl@qti.qualcomm.com; kangk@qti.qualcomm.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2018.2801560

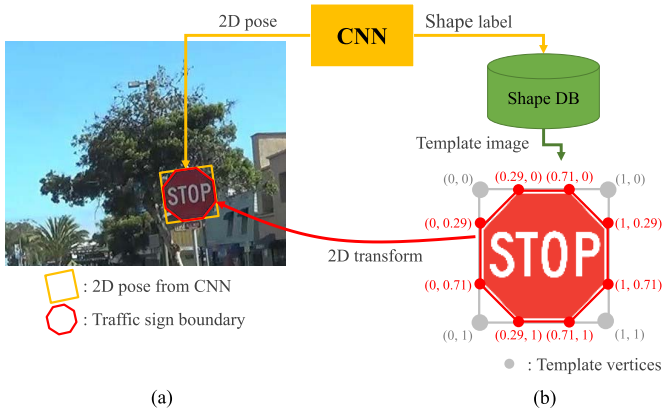


Fig. 1. Boundary estimation using a predicted 2D pose and a shape label: 2D poses and shape labels are predicted from CNN. The template image (b) of the predicted shape label is retrieved from the shape database, and then the boundary corners of the template image are projected to the observed image (a) by the predicted pose.

propagation of CNN which can be accelerated by GPUs. Combining with our efforts to find a base network architecture that provides the best trade-off between accuracy and speed, our precise boundary detection system can be run on mobile platforms with frame rates higher than 7 frames per second (FPS) with affordable traffic sign detection and boundary estimation accuracy.

The rest of paper is organized as follows. Section II reviews previous works on traffic sign detection as well as generic object detection based on CNN. The details of our method are described in Section III including the structure of our detection network, traffic sign boundary estimation using the network output, and training details. In Section IV, we report evaluation results on the proposed method including accuracy and speed, and intensive experiments on speed up of the network. Conclusion and future direction are summarized at the last section.

II. Related Works

A. Traffic Sign Detection

Most of the previous works on traffic sign detection rely on handcrafted image features to identify target signs. Various combinations of features and classifiers are proposed to pursue a robust and fast traffic sign detector, and near one-hundred percent accuracy is achieved on small benchmark dataset such as German Traffic Sign Detection Benchmark (GTSDB) dataset [8]. Since running a complex feature extractor and a classifier is time-consuming, multi-stage cascade architectures composed of fast candidate search and accurate candidate classification have been a typical pipeline of traffic sign detection.

For the early stage of detection pipeline, simple but effective features such as color and edge are used to extract a number of candidate regions. Color is one of the most distinct features of traffic signs, thus many works have utilized color features for fast candidate region search in various forms, such as color probability maps [9], [10] and multiple thresholding [11]. More discriminative features like saliency, textures, and patterns are favored in the middle stages of the pipeline. In [12], saliency features are computed in multiple scales, and then each feature of different scales is tested in cascade classifiers. In [13] and [14], variants of local binary pattern (LBP) are extracted and fed into cascade classifiers. Region-based features like histogram of gradient (HoG), which compute the statistics of primitive features in the region, are widely used in the final stage of the detection pipeline where accurate decision on traffic sign/non-traffic sign is required [9], [12], [15].

Meanwhile, more sophisticated features such as the integral channel features (ICF) [16] or the aggregated channel features (ACF) [17] have also been applied to traffic sign detection systems [18], [19]. The ICF/ACF features have strong discriminative power while being efficiently computed, thus a single classifier shows competitive accuracy compared to the cascaded classifiers with simpler features.

Depending on the implementation, traffic sign detectors based on the handcrafted features can be very fast without any special hardware (*e.g.,* detection module in [19] takes 76ms on PC without GPU), but their accuracy could be severely affected by hyper-parameters and various conditions such as weather, light, or camera properties, as features designed by hand are known to be weaker than features learned by CNN from massive training data [20].

While traffic sign classification has utilized CNN earlier [21] and shown highly accurate recognition results, the use of CNN in traffic sign detection has started recently in [22] and [23]. In these works, the use of CNN is simply to apply CNN classifier to candidates regions to reject non traffic sign candidates, thus their accuracy and efficiency are significantly affected by candidate extraction step which still relies on handcrafted features. More comprehensive reviews and comparisons on the recent advances in traffic sign detection and recognition systems are studied in [24]–[26].

B. CNN for Object Detection

Recently, great advances have been achieved on object detection by CNN [5]–[7]. Besides the discriminative power of CNN on object category classification, the detection networks show the capability of accurate localization by performing *regression* on object location (This location regression will be explained at Section III). Two different architectures of detection networks are currently being developed: direct detection [6], [27] and region proposal based detection [7], [28]. In direct detection, predictions on the position (by regression) and class (by classification) of target objects are directly obtained from convolutional feature layers, resulting in relatively faster run time. On the other hand, region proposal based methods first generate a number of candidate regions regardless of their classes, and then perform prediction on object position and class for each candidate region. By performing regression and classification twice in different stages of the network pipeline, the region proposal based methods pursue more accurate detection, with relatively slower run time than direct detection methods [29]. For the case of traffic sign detection for autonomous driving, direct detection methods

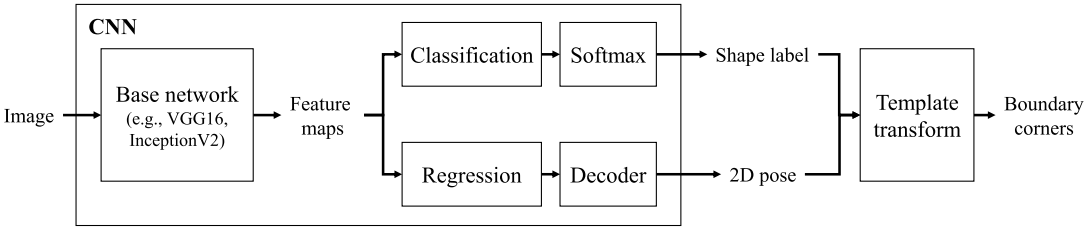


Fig. 2. The overall procedure of the proposed method. Inside CNN, convolutional feature maps are extracted from the input image, and the shape labels and 2D poses for each default box in a grid are predicted from the feature maps. Then we transform corners of template images that corresponds to the shape label using the 2D pose to get the boundary corners for input image.

are arguably more adequate since the latency of detection is important under limited computational resources.

Although most of the recent CNN object detection methods provide accurate bounding box and class label prediction, further processes should follow to obtain precise object boundaries from the predicted bounding boxes. To resolve this issue, in [30], boundaries of traffic signs are simultaneously obtained as segmentation masks by *OverFeat* [31]-style convolutional neural network trained on multiple tasks comprising bounding box detection, mask segmentation, and sign category classification. However, predicting pixel-wise segmentation masks requires intensive computation, resulting in very slow speed of the network. On the other hand, we propose boundary estimation method which does not require pixel-wise segmentation and thus enables fast detection speed.

III. Proposed Method

The overall procedure of the proposed method is illustrated in Figure 2. For the CNN block, any object detection network structure [6], [7], [27], [28] which regresses the position of the target object can be used. In our work, we build the CNN block based on the SSD structure where predictions are directly preformed across multiple feature levels. The main difference of our network with the previous detection networks is what it predicts as output: instead of predicting bounding box coordinates, our network performs *pose estimation*, which can be converted into the boundary estimation of corresponding traffic signs. In the CNN block, an input image is passed to a base network which extracts feature maps using a series of convolution, non-linear activation, and pooling operations. Then from the feature maps, 2D poses and shape class probabilities are estimated by two separated convolutional layers, namely, *pose regression layer* and *shape classification layer*, combined with successive operations that convert the convolution outputs to the 2D pose values and class probabilities, respectively (through Softmax and Decoder in Figure 2). Finally, we use the obtained 2D poses and shape class probabilities to compute boundary corners.

A. Convolutional Feature Extraction

Since pre-trained CNNs on ImageNet classification task [20], [32]–[34] have become dominant as a starting point for fine-tuning CNNs for other vision tasks [35], most of the recently introduced object detection methods also borrow one of these networks for feature extraction upon which detection layers perform bounding box estimation and object category classification. We also choose our base networks among these networks with a notable consideration: since we aim at using our method for real-time vehicle applications, the most important criterion to choose our base network is the accuracy/speed trade-off. For detailed comparison on base network for object detection, refer to [36], and we present experimental results using VGG16 and InceptionV2 base network in Section IV.

Similar to other CNN based object detection methods, our pose regression and shape classification layers perform on top of convolutional feature maps generated from these base networks. Since our implementation adopts SSD structure where detection is performed on multiple feature maps of different spatial resolutions, our pose regression and shape classification layers also spread in multiple layers to reliably detect traffic signs of different scales.

B. 2D Pose Prediction

We first review the bounding box regression used in recent CNN based object detectors [6], [7]. The regression layers, which can be either convolution layers or fully-connected layers, take feature maps from the base network and predicts pose offsets relative to the default box coordinates. Here, a set of default boxes of different aspect ratios and scales is assigned to each unit of the feature maps in the regression layers as shown in Figure 3(a). Since a bounding box can be represented by two 2D coordinates, *i.e.,* (left, top, right, bottom), 4-dimensional vectors can be used to represent regression values. For example, the regression vector can be represented as $(\Delta l, \Delta t, \Delta r, \Delta b) = (l_t, t_t, r_t, b_t) - (l_d, t_d, r_d, b_d)$, where l, t, r, and b are left, top, right, and bottom coordinate of bounding box, respectively, and the subscripts $_t$ and $_d$ denote coordinates in target box and default box. A target box corresponds to a ground truth box in the training stage and a predicted box in the inference stage. In Figure 3(b), two green arrows correspond to $(\Delta l, \Delta t)$ and $(\Delta r, \Delta b)$, and by adding these two vectors to left-top and right-bottom corners of default box respectively, a bounding box can be obtained as a detection result.

In our method, on the other hand, the box regression is generalized to vertex regression to predict 2D poses of targets. The 2D pose of a planar target in an image can be represented as perspective transform of the target, and the simplest way to determine the 2D pose is to obtain four points on the target plane. We predict a 8-dimensional regression vector

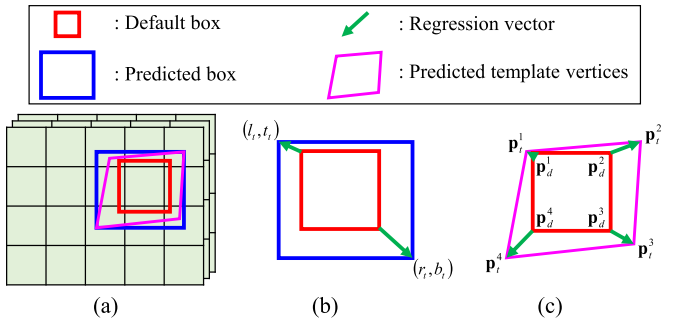


Fig. 3. Box and pose prediction using regression vectors. (a) For every unit of feature maps in the regression layer, we assign default boxes and obtain regression vectors for each of them. (b) In conventional object detection networks, two regression vectors are predicted for bounding box estimation. (c) In our model, four regression vectors are predicted from the network for 2D pose estimation.

of four points to get the 2D pose of the target. The four points need not be actual corner points of the target, and arbitrary predefined locations on the target can be used to determine the 2D pose. To estimate precise boundaries of traffic signs, we use *'template vertices'* of traffic signs (see Figure 1), which are virtual corners determining the minimum bounding quadrilateral of the signs as regression points. Then the regression vector can be defined as

$$\Delta \mathbf{p}^i = \mathbf{p}_t^i - \mathbf{p}_d^i, \quad i = 1, 2, 3, 4, \tag{1}$$

where $\Delta \mathbf{p}^i$ is ith regression vector corresponds to ith target template vertex $\mathbf{p}_t^i$ and vertex of default box $\mathbf{p}_d^i$, as illustrated in Figure 3(c). In the inference stage we can reversely estimate template vertices from the network output $\Delta \mathbf{p}^i$ as

$$\mathbf{p}_t^i = \mathbf{p}_d^i + \Delta \mathbf{p}^i, \quad i = 1, 2, 3, 4. \tag{2}$$

C. Boundary Corner Estimation

Given the 2D pose of a target sign, it is straightforward to determine the boundary of the sign given its template image. First, from the shape classification layer we predict the shape label of the target to determine which template image to use. The output of the shape classification layer, denoted by $\mathbf{s}$, is N-way softmax values where N is the number of shape classes (rectangle, diamond, octagon and background in our experiments). We use one template image for each corresponding shape (for example, the shape label is octagon in Figure 1) where coordinates of its boundary corners are already known as in Figure 4. For convenience, we normalize the coordinates of a template image, *i.e.,* the width and the height of the template image, to be 1. From the predicted template vertices $\{\mathbf{p}_t^i\}_{i=1,2,3,4}$, we can compute a 2D transform matrix $\mathbf{H}$ satisfying the following equation,

$$\tilde{\mathbf{p}}_t^i = \mathbf{H}\tilde{\mathbf{q}}^i, \quad i = 1, 2, 3, 4, \tag{3}$$

where $\tilde{\mathbf{p}}^i$ and $\tilde{\mathbf{q}}^i$ are the homogeneous representations of $\mathbf{p}^i$ and $\mathbf{q}^i$, respectively, and $\mathbf{q}^i$s are the coordinates of template image corners, such that $\mathbf{q}^1 = [0, 0], \mathbf{q}^2 = [1, 0], \mathbf{q}^3 = [1, 1], \mathbf{q}^4 = [0, 1]$. Finally, we transform the boundary corners $\{\mathbf{c}^j\}_{j=1,2,\ldots,M}$ where M is the number of boundary corners in the template image (*e.g.,* $M = 8$ for octagon sign) using the computed transform matrix $\mathbf{H}$ into the image coordinate to obtain the precise boundary $\{\mathbf{b}^j\}_{j=1,2,\ldots,M}$ of the detected traffic sign.

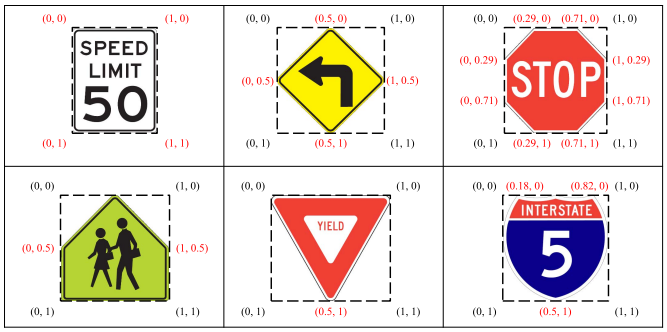


Fig. 4. Examples of traffic sign template images and their boundary corners.

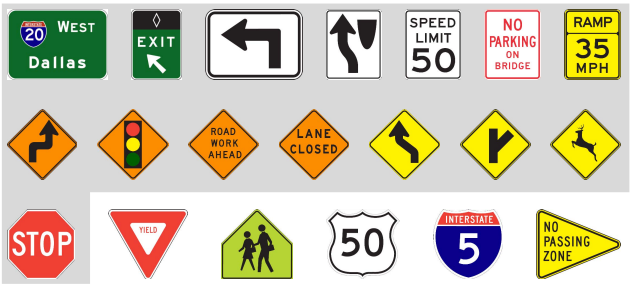


Fig. 5. Examples of U.S. traffic signs categorized by shape. *Top row*: rectangle shape, *Middle row*: diamond shape, *Bottom row*: other shapes. Traffic sign shapes in the shaded area are used in our experiment.

Figure 4 shows examples of template images with various traffic sign shapes. The boundary corners $\{c^i\}$ are indicated by red coordinates. In case of 'triangle' and 'shield' shapes, we can represent the transform matrix $\mathbf{H}$ with an affine matrix, while other shapes having more than 3 corners can be represented by a perspective matrix. For traffic signs with circle shape, which are common in non-U.S. traffic signs, we can use four endpoints of major and minor axes of observed ellipse to compute the perspective transform matrix. In our experiments we only use rectangle, diamond, and octagon shapes since they are most frequent shapes in U.S. traffic signs. In fact, more than 90% of U.S. traffic sign types have rectangle or diamond shapes [37]. Figure 5 shows examples of traffic signs from *California Manual on Uniform Traffic Control Devices* [37] for each shape class.

D. Training

All our models are based on base networks that are pre-trained on ImageNet dataset [38]. The network parameters are fine-tuned during training the entire model. The pose regression and shape classification layers are trained from scratch with 'Xavier' initialization [39]. To train the pose regression and shape shape classification layers, we determine positive and negative samples and use them to compute a loss function. The positive samples correspond to predictions from

default boxes matched with any of ground truth traffic sign, where 'matched' means that the intersection over union (IoU) between the default box and the ground truth target is larger than 0.5. If a default box is not matched with any ground truth target, then the prediction from the default box is regarded as a negative sample.

We use the softmax loss as shape classification loss $L_{shape}(\mathbf{s})$ for all positive and negative samples, and use smooth $l1$ loss [40] as vertex regression loss $L_{vertex}(\Delta\mathbf{p}^i)$ only for positive samples. For negative samples, the vertex regression loss is not assigned. The overall training loss for a single image is the sum of the shape classification loss $L_{shape}(\mathbf{s})$ and the vertex regression loss $L_{vertex}(\Delta\mathbf{p})$ as follows,

$$L_{overall}(\mathbf{s}, \Delta\mathbf{p}^i) = \frac{1}{K_p + K_n}\lambda_s \sum^{K_p+K_n} L_{shape}(\mathbf{s}) + \frac{1}{K_p}\lambda_v \sum^{K_p}\sum_{i=1}^{4} L_{vertex}(\Delta\mathbf{p}^i), \quad (4)$$

where K_p and K_n are the number of positive and negative samples respectively, and λ_s and λ_v are loss weights for the shape classification loss and the vertex regression loss respectively. For notational simplicity, we omit the index of a default box inside each loss function.

Since multiple default boxes of various aspect ratios are assigned to each unit of classification and regression layers, it is usual that the total number of default boxes is significantly larger than the number of positive samples, and most of the default boxes become negative samples. To resolve this severe imbalance between positive and negative samples in an input image, we employ *hard negative mining* technique [6], [41] to select only few of the negative samples with highest classification scores given by the classification layers. In our experiments, we set $K_n = 3K_p$ (K_p is dependent on the number of ground truth objects in the image), and use only top-K_n negative samples. We optimize this objective loss using the stochastic gradient descent method with initial learning rate 10^{-3}, momentum 0.9 and batch size 32. The learning rate is decreased to 10^{-4} after 48,000 iterations, and optimization continues until 60,000 iterations.

E. Dataset Manipulation

For experiment data, we collected 37,079 road scenes containing traffic sings on highway as well as local road around San Diego, California, using a wide field of view camera mounted on a dashboard of a car. We refer to this 'San Diego Traffic Sign' dataset as *'SDTS'*, and the number of signs for each shape is reported at Table I.[1] The dataset is split into two disjoint sets; 33,360 training images and 3,719 test images which are captured at different paths. The collected images have slight radial distortion, but ground truth annotation, network training and testing are done without correcting the distortion. For each image, boundary corners of traffic signs as well as their shape labels are annotated.

[1]There are several benchmark dataset for traffic sign detection such as GTSDB dataset [8] for German traffic sign and LISA dataset [42] for U.S. traffic sign, but we can not use these datasets since they only have bounding box annotations.

Fig. 6. Examples of unusable images for training. *Top*: traffic sign touching image border. *Bottom*: occluded traffic sign.

TABLE I
NUMBER OF SIGNS FOR EACH SHAPE IN *'SDTS'* TRAINING AND TEST SETS

	Training set	Test set	Total
Number of images	33,360	3,719	37,079
Number of signs	57,692	6,324	64,016
Rectangle	45,961	5,188	51,149
Diamond	10,772	1,093	11,865
Octagon	959	43	1,002

To train pose regression layers using the annotated data, we convert the annotated boundary corners $\{\mathbf{b}^j\}$ into template vertices $\{\mathbf{p}_t^i\}$ by calculating a transform matrix $\mathbf{H}$ from $\{\mathbf{b}^j\}$ and boundary corners $\{\mathbf{c}^j\}$ in the template image, and then transforming image corners $\{\mathbf{q}^i\}$ of the template image using the matrix $\mathbf{H}$ to obtain $\{\mathbf{p}_t^i\}$.

For more effective training, we manipulate the training data with pruning, augmentation, and hard negative mining as follows.

1) Pruning Training Data: In case of occluded or partially observable traffic signs as shown in Figure 6, it is difficult to annotate accurate boundary corners of these signs. Since using the vertex regression loss from these samples may degrade the vertex prediction accuracy, we decided not to use them during training. In order not to modify the overall loss function (4), we exclude images that contain at least one occluded or partially observable traffic signs, instead of not assigning vertex regression loss to those samples. Although those partially observable traffic signs are not used in training, the trained detector can robustly estimate boundary of such signs as shown in Figure 12.

2) Data Augmentation: To enhance the robustness of our detector to various appearance changes of traffic signs, we populate the training data with data augmentation techniques. We applied image cropping and perspective transform to training data: image cropping helps the detector be less sensitive to the position and size of traffic signs, and perspective transform makes the detector robust to view point changes. Especially, as the camera gets closer to a traffic sign, the sign

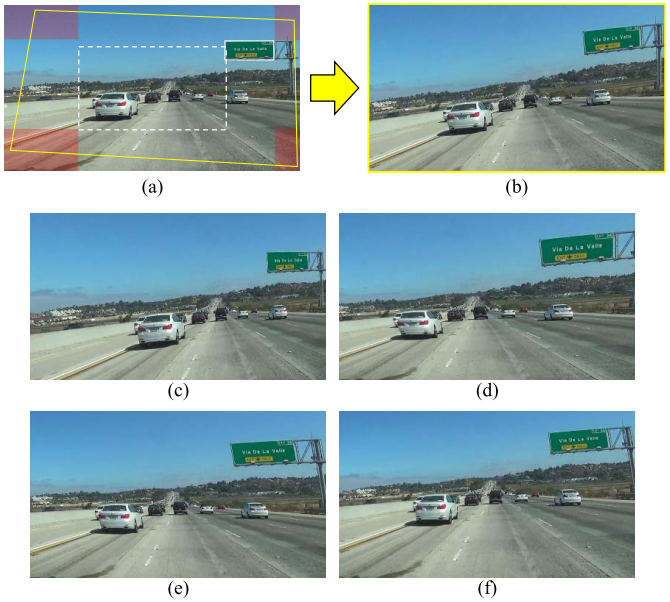


Fig. 7. Perspective data augmentation for close and large sign targets. In the original image (a), four corner points are randomly sampled from each red regions. The red regions are determined by union of half center region (dotted white rectangle) and sign target (solid white rectangle). The yellow quadrilateral determined by the four corner points are then transformed to the augmented image with the original image size as shown in (b). Images in (c)∼(f) are examples of augmented images from image the same image in (a). We generate 10 augmented images from one original image.

generally moves toward the image boarders and has severe perspective distortion compared to distant signs. However, the number of close sign samples are limited compared to distant sign samples, because objects shift faster as they come close to the camera. We compensate this unbalance of samples between distant and close traffic signs by augmenting close signs with perspective transform. Only for images containing at least one large sign (about 620 images out of 33,360 training images), we generate randomly transformed duplicates for each of the images using perspective transform as shown in Figure 7. If a transformed image has a sign at the image border, we discard it and regenerate a new transformed image so that they do not contain partially observable traffic signs. In our experiment, we generated 10 augmented images from each original image, resulting in 6,200 augmented images in total.

3) Hard Negative Mining: In addition to image-level hard negative mining described in Section III-D, we employ dataset-level hard negative mining in an aggressive manner to make our detector more robust to false positives. To our training data we add additional images that do not contain any positive samples and have confusing objects of traffic signs. This is a notable difference compared to previous works [6], [7] in that they only use training images that contain at least one positive example. Concretely, we first train our model on training images all of which include traffic signs, and run the trained model on training images of confusing objects to find hard negatives. We sort these hard negatives and add the top 5,000 images to the training data in addition to the initial 33,360 training images. Our final models used in our experiments were trained on this extended training data. Since traffic signs and frequently confusing objects (advertisement signs, some types of trees, etc.) are not often collocated, this strategy helps our models benefit from hard negative mining effectively. We confirmed in our experiments that adding these negative images significantly reduce false positives.

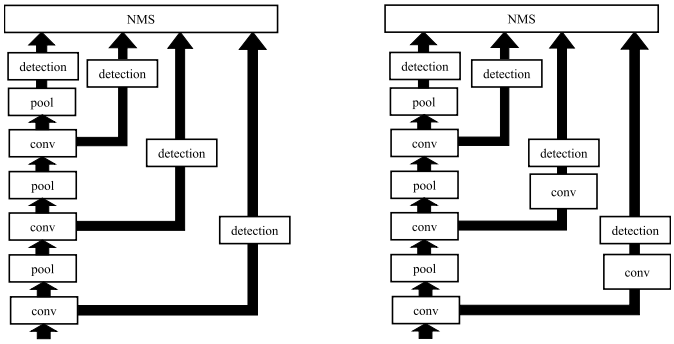


Fig. 8. A graphical depiction of inserting additional convolutional layers into lower detection paths. Example model diagrams without additional convolutional layers (left) and with additional convolutional layers (right).

IV. Experiments

A. Network Implementation

The proposed pose regression layer can be combined with various modern object detection networks such as Faster R-CNN [7] and SSD [6]. We particularly chose to implement our models on top of the SSD framework since it is significantly superior to Faster R-CNN in terms of computational efficiency while being competitive in accuracy. As reviewed in Section II, SSD generates position and class predictions directly from feature layers instead of using intermediate region proposals. To cover multiple scales of target objects, predictions of SSD are obtained from multiple feature layers of different resolutions. We use 6 feature layers of different resolutions (1/8, 1/16, 1/32, 1/64, 1/128, and 1/256 size of the network input) as detection sources, and we select the last layer among layers of the same resolution in the base network as the source layer of that resolution. The default boxes are defined with different scales and aspect ratios (1, 2, 3, 1/2 and 1/3 in our experiments) for each source layer, and redundant predictions from all the source layers are filtered out by non-maximum suppression (NMS) according to classification confidence.

Upon this SSD structure, we modify the network structure to strengthen the predictions in lower layers: we insert additional convolutional layers between the pose regression/shape classification layers and the convolution layers on which those layers perform prediction, to compensate their relatively shallower depths from input. A graphical depiction of this approach is in Figure 8. We apply this extension for two lower prediction layers (1/8 and 1/16 scales) since higher layers already have enough depths and receptive field sizes. Each layer has 192 and 256 output channels with 3×3 kernels respectively. Given the fact that small traffic signs appear more frequently (refer to section III-E.2), this extension is helpful to improve detection accuracy as discussed in section IV-E.

B. Experimental Setup

All the training process was conducted using *Caffe* framework [43]. For testing, we implemented the system using Snapdragon Neural Processing Engine (SNPE) SDK which provides an optimized implementation of CNN forward propagation on the Qualcomm Snapdragon ™ 820A processor[2] with GPU utilization. For both training and testing, we used RGB channel color images, and different image resolutions for base network are tested to evaluate trade-off between accuracy and speed.

C. Accuracy Evaluation

Before presenting the evaluation of speed optimized network (Section IV-D), we first present the accuracy evaluation of our basic detection network using VGG16 as base network.

1) Evaluation Protocol: We computed two measures for traffic sign detection accuracy: the mean average precision (mAP) and the average vertex error (AVE). The mAP is to measure detection rates and the average vertex error is to measure the accuracy of boundary estimation of detected signs. For each boundary estimation that matches a ground truth boundary ("matches" corresponds to the condition that IoU between the detected box and the matched ground truth box is above a predefined threshold), the vertex error v_err is computed by the average distance between the estimated boundary corners b_e^j and the ground truth boundary corners b_g^j using the following equation:

$$v\_err = \frac{1}{M}\sum_{j=1}^{M} \|b_e^j - b_g^j\|, \tag{5}$$

where M is the number of corners in the traffic sign.

2) 'GTSDB' Test Dataset: Since our method requires polygon annotations for training as well as evaluation of average vertex error, we cannot evaluate the boundary detection accuracy of our method on existing benchmark datasets. However, it is possible to evaluate only the detection accuracy with respect to bounding box, thus we compare the accuracy of ours (trained on box annotations of GTSDB training data) with the state-of-the-art methods on GTSDB test dataset as summarized in Table II. We can see that our detector using VGG16 base network gives the best accuracy with respect to average box overlap (AO) for all traffic sign categories, with competitive accuracy with respect to area under precision-recall curve (AUC).

3) 'SDTS' Test Dataset: For accuracy evaluation, we used test images from *'SDTS'* dataset which comprises 3,719 images with 6,324 traffic sign annotations. We calculate the mAP with varying IoU values, and the results are plotted in Figure 9. As shown in the figure, both models achieve above 0.8 mAP at 0.5 IoU, and even obtain near 0.8 mAP at 0.7 IoU. This shows that our methods are able to detect traffic signs with precise accuracy. More comparison between two base networks will be discussed in Section IV-D1.

As reported in Table I, we have a very unbalanced number of training samples for each sign shape. To check how does this data unbalance affect the accuracy of detection, we measure accuracy numbers for each shape independently, and the results are reported at Table III. Interestingly, we can see that the accuracy of the rectangle shape is worse than other shapes although it has much more training samples. One plausible explanation of the worse detection accuracy on the rectangle shape might be the wide variety of appearances of

[2]Qualcomm Snapdragon ™ is a product of Qualcomm Technologies, Inc.

TABLE II

BOUNDING BOX ACCURACY COMPARISON ON GTSDB (%)

Method	Prohibitive		Danger		Mandatory	
	AUC	AO	AUC	AO	AUC	AO
wgyHIT501 [15]	**100**	90.08	99.91	86.39	**100**	79.41
visics [44]	**100**	88.22	**100**	87.03	96.98	89.55
LITS1 [45]	**100**	86.91	98.85	86.34	92	89.49
BolognaCVLab[46]	99.98	84.95	98.72	86.78	95.76	86.19
Ours	99.89	**91.93**	99.93	**91.87**	99.16	**91.47**

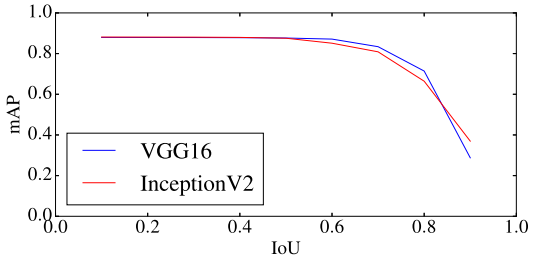


Fig. 9. mAP versus IoU plot by detectors based on VGG16 and InceptionV2, respectively. Our optimized InceptionV2-based detector shows comparable detection rate with VGG16-based detector for most range of IoU.

TABLE III

ACCURACY EVALUATION ON TRAFFIC SIGN SHAPE WITH VGG16 BASE NETWORK

Shape	# GTs	Precision	Recall	AVE
Rectangle	6444	0.895	0.705	2.590
Diamond	1124	0.887	0.877	2.089
Octagon	47	0.913	0.894	2.280
Total	7615	0.894	0.732	2.499

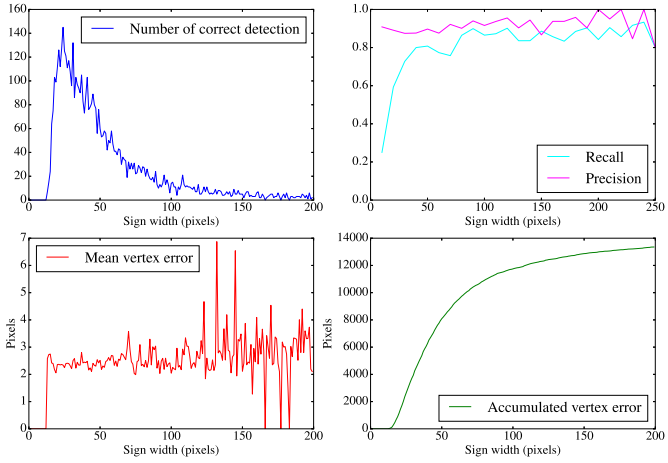


Fig. 10. Detection accuracy over traffic sign sizes. The size of signs is determined by the width of the sign in 1280 × 720 resolution.

Fig. 11. Examples of detection results. Green, orange, and red colors indicate rectangle, diamond, and octagon shape, respectively, and white rectangles indicate bounding boxes. *Left column*: images from local road scenes. *Right column*: images from highway scenes. *Top row*: All signs are detected correctly. *Middle row*: Results include false negatives: small signs are not detected (locations of false negatives are indicated by yellow rectangles). *Bottom row*: Results include false positives: a rectangular pattern is detected as a traffic sign in a local road scene, and an electronic sign board is detected in a highway scene.

the rectangular traffic signs, as well as the existence of many confusing objects of similar appearance such as advertisement signs.

Examples of detection results including failure cases are shown in Figure 11, and the video is available at the web.[3] In this experiment, input images to the base network have the resolution of 800×450 pixels, and signs whose side length is smaller than 13 pixels are ignored (Figure 11 middle). As shown in the examples, our model is able to detect and estimate precise boundaries of traffic signs of various sizes and shapes. Most of the false positives are caused by objects of similar appearance, as in the bottom of the figure where a chimney or a electronic sign board is detected as a rectangular sign. With more training samples, especially various negative samples, these false positives are reduced.

4) Comparison With Segmentation-Based Methods: We compare the boundary estimation accuracy of our method with a segmentation-based method combined with bounding box detection. For the segmentation-based method, same detection results are used with our method except the results for the segmentation-based method are represented by bounding boxes (circumscribed rectangles of template vertices). For each bounding box, a patch is cropped with boundary padding, and then the patch is passed to the segmentation network (based on [2]) trained on cropped traffic sign patches to generate binary masks. On the boundary of the mask, polygon fitting and simplification is applied to obtain final boundary corner output. As shown in Figure 12, our method is more robust to partial occlusion, cluttered background, and close signs which can not be directly handled by segmentation algorithm: since the segmentation-based method cannot restrict its output masks to fit to traffic sign shapes, an additional step for boundary fitting is required to refine the masks. We compare the average vertex errors of the two methods: in VGG-16 based models, the segmentation-based method gives 3.2917 pixels error while our method gives 2.7014 pixels error.

For speed comparison, we run both methods on PC with *Titan X* GPU. Our method takes 126ms per image, while the segmentation-based method takes 189ms per image excluding computation time for boundary fitting (106ms for SSD to detect bounding boxes and 83ms for the segmentation network). Depending on used boundary fitting algorithm, overall computation time of the segmentation-based method can be much increased.

5) Sign Boundary Refinement: Though our models provide robust boundary estimation for traffic signs, we observed that the accuracy can be improved further with an image-based local refinement by using the edge or gradient of sign boundaries. In the refinement process, we compute the gradient image for the patch of a detected traffic sign, then estimate an optimal 2D transform that minimizes the weighted distance between the computed gradient image and the predicted boundaries from our network. The 2D transform can be either

[3]Qualcomm ® drive data platform, https://www.qualcomm.com/news/onq/2017/02/03/zoom-what-powers-qualcomms-drive-data-platform

Fig. 12. Comparison of boundary estimation between our method and segmentation-based method. *Top row*: occluded sign. *Middle row*: cluttered background. *Bottom row*: close signs.

TABLE IV
DETECTION ACCURACY BY A DIFFERENT NUMBER OF TRAINING IMAGES. VGG16 IS USED FOR THE BASE NETWORK

# training images	mAP @0.5 IoU	AVE
5,000	0.540	3.182
10,000	0.892	2.882
15,000	0.900	2.658
20,000	0.884	2.620
25,000	0.887	2.570
30,000	0.887	2.600

perspective or affine, and in our experiments affine transform gives faster and more robust refinement results. Since the goal of this step is local refinements, we discard the refinement result if it is far from the original prediction given by the network. By the refinement, the average vertex error is reduced from 2.679 to 2.256, while it takes 1.2ms per image on PC.

6) Volume of Training Data: One drawback of CNN-based detection networks is that it requires a large number of training images. Although we prepared more than 44,000+ images (including 33,000+ images with traffic sign annotations, 5,000 negative-only images, and 6,000+ augmented images) for training, it is not clear that this amount of data is sufficient and more data can improve the accuracy. Therefore, we experimented with different number of training images and evaluated the accuracy of the trained networks. As reported at Table IV, the detection accuracy marginally increases with larger data volumes, when the number of training images is larger than 15,000. The effect of data augmentation is reported at Table VIII.

D. Speed Optimized Network

Though our CNN based traffic sign detector shows reliable accuracy, the heavy computational demands from many layers in our network hinder our models from running on low-power devices for autonomous driving and mapping. For example, our best-performing model relies on VGG16 network, which requires more than 100 billion multiply-accumulate operations (MAC) in a single forward pass for 800 $\times$ 450 resolution input. On GPU-supported Snapdragon ^{TM} 820A processor, this amount of computation only provides less than one FPS, surely far from a minimal requirement for our target applications. To achieve practical FPS, we apply two approaches both of which directly reduce the number of required MAC of our models: using lighter base networks and reducing the input image resolution.

1) Lighter Base Network: Recent advances in deep CNN have shown that increasing the depth of neural networks increases accuracy [33], [34]. However, the accuracy gain is relatively marginal compared to the increased computational cost, and this prevents adopting recent deep CNN to low-power devices where computational capacity is clearly limited. To make our models accommodate to low-power devices, we replace the base network from VGG16 to a computationally lighter network. We select InceptionV2 [47] network as an alternative since it has shown to provide a good trade-off of accuracy and speed [36]. InceptionV2 requires about 14 billion MAC in one forward pass for 800 $\times$ 450 resolution input, less than one seventh of MAC for VGG16.

Replacing the base network from VGG16 to InceptionV2 leads significant increase in FPS, as shown in Table V. Nonetheless, it just provides less than 3 FPS, still far from realizing our systems practical. To further speed up our network, we decrease MAC of InceptionV2 network by removing its several higher layers. The motivation of this approach is that our target objects (*i.e.,* traffic signs) are relatively simple in their shapes and structures compared to generic object classes, thus the higher layers have less importance since they are known to be needed to learn hierarchical structures of object parts and complex shapes. To verify this approach, we conducted experiments on using only some lower part of InceptionV2 network for our task. Table V shows the results. As expected, removing most of upper layers does not significantly affect detection accuracy. We found that removing all upper layers from 'inception4b' module reduces almost half of MAC yet degrades detection accuracy marginally.

2) Input Image Resolution: Since our network is fully-convolutional, MAC required for our model are directly proportional to the number of input pixels. This means that the input image resolution is a key factor of overall computational cost. To reduce the input resolution while keeping detection accuracy as much as possible, we conducted experiments on the trade-off between speed controlled by input resolution and accuracy measures (mAP and vertex errors). Table VI shows that 533 $\times$ 300 input shows near 5 FPS with competitive accuracy to higher resolution settings, providing a reasonable trade-off between speed and accuracy. Sacrificing little more accuracy leads to more than 7 FPS by 400 $\times$ 225 resolution input. With a robust object tracking method, our traffic sign detection of around 5 FPS could be a reasonable option for near real-time traffic sign detection systems which is one of the key components of autonomous driving and mapping.

TABLE V

PERFORMANCE TABLE. MAC NUMBERS ARE CALCULATED BASED ON 224 × 224 INPUT RESOLUTION AND FOR BASE NETWORKS ONLY. FPS NUMBERS ARE CALCULATED WITH 800 × 450 INPUT

Base Network	MAC (million)	mAP	AVE	FPS
VGG16	15470	88.4	2.746	0.2
InceptionV2	2018	86.8	2.928	1.5
InceptionV2 cut4b	1270	86.4	3.051	2.1
InceptionV2 cut4a	1145	86.5	3.126	2.4
InceptionV2 cut3c	1025	85.2	3.212	2.5
SqueezeNet	390	75.4	3.818	4.3

TABLE VI

PERFORMANCE INPUT RESOLUTION WITH 'INCEPTIONV2 CUT4A' OPTIMIZED MODEL

Input Resolution (# pixels)	mAP	AVE	FPS
800 × 450 (360,000)	86.5	3.126	2.4
600 × 360 (201,600)	85.4	3.4425	4.2
533 × 300 (159,900)	85.2	3.5824	5.3
400 × 255 (90,000)	82.8	3.7238	7.3

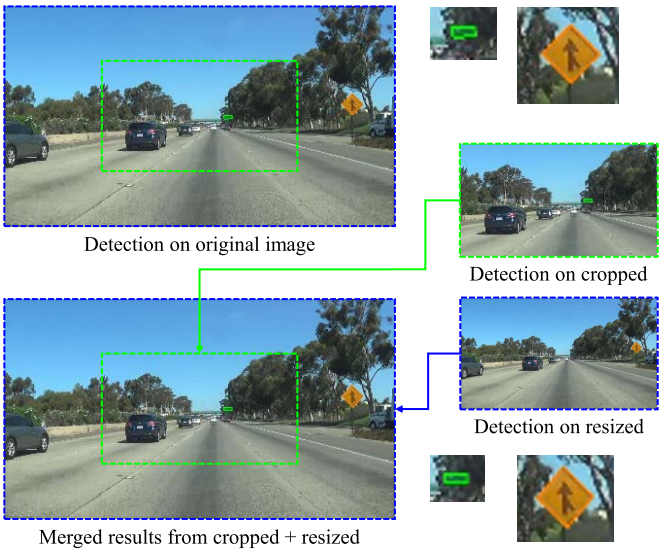


Fig. 13. Detection on cropped and resized images gives comparable results with detection on original images but with much less time.

3) Detection Speed Up by Input Cropping: Besides the choice of base networks and input image resolution which are significant factors in accuracy/speed trade off, we applied a simple trick to improve the speed of our models by leveraging a characteristic of traffic scene. Different from generic object detection, there is a strong correlation between target sizes and positions for traffic scenes. Distant targets usually appear near the image center, and they are relatively difficult to detect due to their small sizes. On the other hand, close targets appear near the image border with large size. Therefore, we can run two detectors with input images of difference sizes: one for a center-cropped image and the other for a half-resized image as shown in Figure 13. The center-cropped image has the same resolution with the original image and distant targets can be detected without accuracy degradation. On the other hand, close signs can be detected from the half-resized image with small accuracy degradation compared to the original image.

TABLE VII

RUNNING TWO NETWORKS FOR CROPPED AND RESIZED IMAGES. (TIME ON PC WITH GPU, WITH VGG16 BASE NETWORK)

Model	Recall	Precision	AVE	time (ms)
Original image	0.606	0.913	2.718	126.7
Crop & resize	0.610	0.841	2.8839	79.0

TABLE VIII

EFFECTS OF DESIGN CHOICES AND COMPONENTS ON SSD PERFORMANCE

	SSD InceptionV2 cut4a 533 × 300					
data augmentation		✓	✓	✓	✓	✓
include 2, 1/2 boxes	✓	✓	✓	✓	✓	✓
include 3, 1/3 boxes				✓	✓	✓
include 4, 1/4 boxes					✓	
additional conv			✓	✓	✓	
additional conv (light)						✓
mAP	83.3	84.4	85.3	**86.5**	86.2	86.3
AVE	3.241	3.176	3.152	3.126	**3.097**	3.131
MAC (million)	**5474**	**5474**	6395	6472	6615	5905

Although we run two detectors, the total number of pixels becomes half than the original image, which results in half computation time for CNN compared to detection with the original image. In practice, including additional steps such as image pre-processing and aggregating detection outputs from the two detector, the total running time of the two detector model is 2/3 the time of the single detector model. The accuracy comparison is reported in Table VII.

E. Model Analysis

We explored various SSD design choices to discover better architectures for our task. All the experiments are based on the 'InceptionV2 cut4a' base network and 533 × 300 input resolution. Table VIII shows the results. The data augmentation by perspective transform improves accuracies on all model configurations, by 1.1% of mAP. Choosing aspect ratio for default boxes affects accuracy, and the best model was obtained from five aspect ratios of 1, 2, 3, 1/2, and 1/3. Finally, adding additional convolutional layers to increase the number of convolutional layers on the lower two detection paths improves accuracy by 0.9% of mAP. With the results in Table V and Table VI, it can be seen that the vertex errors are more sensitive to input resolutions and base networks. The MAC of models varies marginally except the additional convolutional layers settings. This is caused by the number of feature maps in the additional layers (192 and 256, respectively). The increase of computational cost can be reduced by using a smaller number of feature maps in the added convolutional layers with marginal accuracy degradation as indicated in the right most column. The lighter layers have half the number of feature maps (96 and 128, respectively), reducing the MAC of the added convolutional layers significantly.

F. Mapping Accuracy

Finally, we compare the traffic sign mapping accuracy between conventional bounding box detection and our precise boundary detection. If there is no geometric change of a

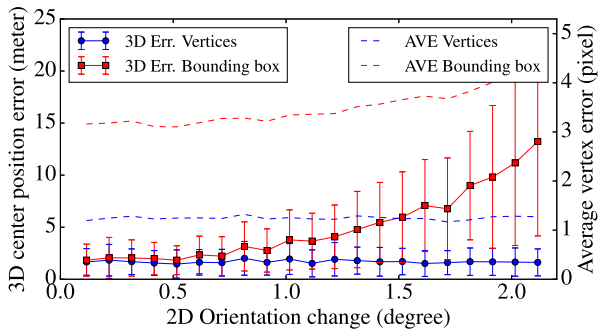


Fig. 14. Comparison of mapping accuracy from bounding boxes and vertices. *Horizontal axis*: 2D sign orientation changes in degree across two frames. *Vertical axis left*: errors of reconstructed 3D sign positions (3D Err.) using observations from two frames. The error plots are represented by averages and standard deviations from 100 simulation results with different observation noise. *Vertical axis right*: Average vertex error (AVE) of the sign observed at the second frame. For bounding box detection, AVE is computed by the average distance between the ground truth vertices and their corresponding corners at the bounding box.

sign (*e.g.,* rotation and perspective distortion) across multiple frames for mapping, then bounding box detection and our boundary detection will produce the same mapping accuracy. On the other hand, when we use bounding box detection instead of boundary prediction, even a small geometric change across frames can cause significant mapping error. Figure 14 shows the simulated comparison of the mapping accuracy between bounding box detection and our boundary estimation, with varying 2D sign orientation changes induced by in-plane camera rotation (rotation around z-axis).

In this simulation, we synthesize images of 1280×720 resolution that emulate scenes from a forward-moving camera observing a rectangular sign. Concretely, we generate pairs of synthetic images each of which consists of two images of a sign observed from different camera positions. For all image pairs, the camera positions (T_x, T_y, T_z) and $(T_x, T_y, T_z + \Delta)$ that simulate forward movement of distance Δ are fixed.[4] The camera rotation around z-axis is 0 for the first frame and θ_z ($\theta_z \in [0, 2]$ degrees in Figure 14) for the second frame, while rotation around other axes remains constant, which results in 2D orientation change between two frames. We project the four corners of the sign into the 2D image plane to obtain 2D vertex coordinates as observation, and add Gaussian perturbation (variance of 1 pixel) to each of the four corners as observation noise. The bounding box observation is defined by the circumscribed rectangle of the four vertices.

We triangulate each pair of 2D observations using Direct linear transformation (DLT) [48], and then measure the sign position error by the Euclidean distance between the centers of the estimated and ground truth 3D corners. Figure 14 shows the simulation results on 2D orientation changes caused by varying camera rotation around z-axis. In the results we can see that mapping based on bounding box detection yields larger errors as the 2D orientation change increases, while mapping based on boundary estimation shows accurate and stable estimation regardless of the 2D orientation change. This indicates that boundary estimation is crucial for robust mapping of 3D objects from 2D observations.

[4]We set T_x, T_y, T_z and Δ as 0, 1, 15 and 5 meters, respectively, and the focal length and principal point of the camera are set to 2000 and the image center, respectively. We set the center of the 3D sign to $(-7, 7, 25)$, and the size of the sign is set as 3×2 meters. The normal direction of the sign is set to be parallel with the camera z-axis.

V. Conclusions

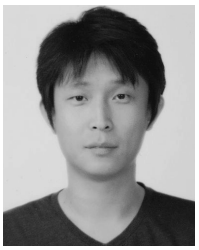
In this paper, we proposed the efficient traffic sign detection method where locations of traffic signs are estimated together with their precise boundaries. To this end, we generalized the object bounding box detection problem and formulated an object pose estimation problem, and the problem is effectively modeled using CNN based on the recent advances in object detection networks. The estimated pose of traffic signs is used to transform the boundary of traffic sign templates into the input image plane, providing precise boundaries with high accuracy. To achieve practical detection speed, we explored the best-performing base networks and pruned unnecessary layers of the network by considering the characteristics of traffic signs. In addition, by optimizing the resolution of network input for the best trade off between speed and accuracy, our detector can run with frame rate of 7 FPS on low-power mobile platforms.

Future direction of our method is summarized as follows. For better accuracy or speed of detection, we can adopt the latest architectures for object detection such as feature pyramid network [49] and multi-scale training [50]. Since the base network is a dominant factor in speed-accuracy trade off, developing the base network specialized for traffic sign detection, instead of using the existing base network for generic object detection, is worth to study. Finally, the proposed method can be applied not only to traffic sign but also to any other planar objects having standard shapes.

References

[1] O. Dabeer *et al.*, "An end-to-end system for crowdsourced 3D maps for autonomous vehicles: The mapping component," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2017, pp. 634–641.
[2] V. Badrinarayanan, A. Kendall, and R. Cipolla. (2015). "SegNet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling." [Online]. Available: https://arxiv.org/abs/1505.07293
[3] J. Uhrig, M. Cordts, U. Franke, and T. Brox. (2016). "Pixel-level encoding and depth layering for instance-level semantic labeling." [Online]. Available: https://arxiv.org/abs/1604.05096
[4] G. Lin, C. Shen, A. van den Hengel, and I. Reid. (2016). "Exploring context with deep structured models for semantic segmentation." [Online]. Available: https://arxiv.org/abs/1603.03183
[5] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587.
[6] W. Liu *et al.*, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 21–37.
[7] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.
[8] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, and C. Igel, "Detection of traffic signs in real-world images: The german traffic sign detection benchmark," in *Proc. Int. Joint Conf. Neural Netw.*, Aug. 2013, pp. 1–8.
[9] Y. Yang, H. Luo, H. Xu, and F. Wu, "Towards real-time traffic sign detection and classification," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 7, pp. 2022–2031, Jul. 2016.
[10] Y. Yang and F. Wu, "Real-time traffic sign detection via color probability model and integral channel features," in *Proc. Chin. Conf. Pattern Recognit.*, Nov. 2014, pp. 545–554.

[11] A. Gudigar, C. Shreesha, U. Raghavendra, and U. R. Acharya, "Multiple thresholding and subspace based approach for detection and recognition of traffic sign," *Multimedia Tools Appl.*, vol. 76, no. 5, pp. 6937–6991, 2017.
[12] D. Wang, S. Yue, J. Xu, X. Hou, and C.-L. Liu, "A saliency-based cascade method for fast traffic sign detection," in *Proc. Intell. Vehicles Symp.*, Jun. 2015, pp. 180–185.
[13] C. Liu, F. Chang, and C. Liu, "Occlusion-robust traffic sign detection via cascaded colour cubic feature," *IET Intell. Transp. Syst.*, vol. 10, no. 5, pp. 354–360, 2015.
[14] C. Liu, F. Chang, and Z. Chen, "Rapid multiclass traffic sign detection in high-resolution images," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 6, pp. 2394–2403, Dec. 2014.
[15] G. Wang, G. Ren, Z. Wu, Y. Zhao, and L. Jiang, "A robust, coarse-to-fine traffic sign detection method," in *Proc. Int. Joint Conf. Neural Netw.*, Aug. 2013, pp. 1–5.
[16] P. Dollár, Z. Tu, P. Perona, and S. Belongie, "Integral channel features," in *Proc. Brit. Mach. Vis. Conf.*, 2009, pp. 1–11.
[17] P. Dollár, R. Appel, S. Belongie, and P. Perona, "Fast feature pyramids for object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 8, pp. 1532–1545, Aug. 2014.
[18] A. Møgelmose, D. Liu, and M. M. Trivedi, "Detection of U.S. traffic signs," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 6, pp. 3116–3125, Dec. 2015.
[19] Y. Yuan, Z. Xiong, and Q. Wang, "An incremental framework for video-based traffic sign detection, tracking, and recognition," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 7, pp. 1918–1929, Jul. 2017.
[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[21] D. Cireşan, U. Meier, J. Masci, and J. Schmidhuber, "Multi-column deep neural network for traffic sign classification," *Neural Netw.*, vol. 32, no. 2, pp. 333–338, 2012.
[22] Y. Wu, Y. Liu, J. Li, H. Liu, and X. Hu, "Traffic sign detection based on convolutional neural networks," in *Proc. Int. Joint Conf. Neural Netw.*, Aug. 2013, pp. 1–7.
[23] R. Qian, B. Zhang, Y. Yue, Z. Wang, and F. Coenen, "Robust chinese traffic sign detection and recognition with deep convolutional neural network," in *Proc. Int. Conf. Natural Comput.*, Aug. 2015, pp. 791–796.
[24] A. Gudigar, C. Shreesha, and U. Raghavendra, "A review on automatic detection and recognition of traffic sign," *Multimedia Tools Appl.*, vol. 75, no. 1, pp. 333–364, 2016.
[25] S. B. Wali, M. A. Hannan, A. Hussain, and S. A. Samad, "Comparative survey on traffic sign detection and recognition: A review," *Przegld Elektrotech.*, vol. 91, no. 12, pp. 38–42, 2015.
[26] P. Saxena, N. Gupta, S. Y. Laskar, and P. P. Borah, "A study on automatic detection and recognition techniques for road signs," *Int. J. Comput. Eng. Res.*, vol. 5, no. 12, pp. 24–28, 2015.
[27] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 779–788.
[28] Y. Li, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 379–387.
[29] J. Huang *et al.* (2016). "Speed/accuracy trade-offs for modern convolutional object detectors." [Online]. Available: https://arxiv.org/abs/1611.10012
[30] Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li, and S. Hu, "Traffic-sign detection and classification in the wild," in *Proc. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 2110–2118.
[31] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "OverFeat: Integrated recognition, localization and detection using convolutional networks," in *Proc. Int. Conf. Learn. Represent.*, 2014, pp. 1–16.
[32] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–14.
[33] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1–9.
[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.
[35] J. Donahue *et al.*, "DeCAF: A deep convolutional activation feature for generic visual recognition," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 647–655.
[36] J. Huang *et al.* (2016). "Speed/accuracy trade-offs for modern convolutional object detectors." [Online]. Available: https://arxiv.org/abs/1611.10012
[37] *California Manual on Uniform Traffic Control Devices*, California State Transp. Agency and Dept. Transp., Sacramento, CA, USA, 2014.
[38] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.
[39] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.
[40] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 1440–1448.
[41] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *Proc. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 761–769.
[42] A. Møgelmose, M. M. Trivedi, and T. B. Moeslund, "Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 4, pp. 1484–1497, Dec. 2012.
[43] Y. Jia *et al.* (2014). "Caffe: Convolutional architecture for fast feature embedding." [Online]. Available: https://arxiv.org/abs/1408.5093
[44] M. Mathias, R. Timofte, R. Benenson, and L. Van Gool, "Traffic sign recognition—How far are we from the solution?" in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Aug. 2013, pp. 1–8.
[45] M. Liang, M. Yuan, X. Hu, J. Li, and H. Liu, "Traffic sign detection by ROI extraction and histogram features-based recognition," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Aug. 2013, pp. 1–8.
[46] S. Salti, A. Petrelli, F. Tombari, N. Fioraio, and L. Di Stefano, "A traffic sign detection pipeline based on interest region extraction," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Aug. 2013, pp. 1–7.
[47] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.
[48] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[49] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. (2016). "Feature pyramid networks for object detection." [Online]. Available: https://arxiv.org/abs/1612.03144v1
[50] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 6517–6525.



Hee Seok Lee received the B.S. and Ph.D. degrees in electrical engineering from Seoul National University, Seoul, South Korea, in 2006 and 2013, respectively. During his Ph.D. study, he was a Research Assistant with the Computer Vision Laboratory. He is currently with the Qualcomm Technologies Inc., Seoul, South Korea. His research interests are in vision-based SLAM and 3-D reconstruction, visual tracking, and object detection.

Kang Kim received the B.S. degree in computer science from Yonsei University, Seoul, South Korea, in 2007 and the M.S. degree in computer science from KAIST, Daejeon, South Korea, in 2009. He is currently with the Qualcomm Technologies Inc., Seoul, South Korea. His research interests include object detection, semantic segmentation, and scene text recognition using deep learning.