Refining Deep Vehicle Detectors for Autonomous Driving

Journal:	*IEEE Transactions on Intelligent Vehicles*
Manuscript ID	T-IV-16-08-0047
Manuscript Type:	Full Length Article
Date Submitted by the Author:	25-Aug-2016

Complete List of Authors:	Rajaram, Rakesh; UC San Diego, Electrical and Computer Engineering Ohn-Bar, Eshed; University of California San Diego, Electrical and Computer Engineering Trivedi, Mohan; University of California San Diego, CVRR-LISA

Keywords:	computer vision, pattern recognition, machine learning, Image Processing, Neural Networt, ITS, Machine Vision, Robot vision, mobile robotics

Abstract:	Robust and accurate object detection is an essential component of an intelligent vehicle. In particular for on-road applications, localization quality of objects in the image plane is important for accurate distance estimation and consequently safe and smooth trajectory planning. In this paper, we investigate a novel strategy to improve localization accuracy of detected objects, in particular on-road vehicles, using a deep convolutional neural network. An iterative region-of-interest pooling framework is proposed for predicting increasingly tight object boxes. The method is shown to converge within 2-3 iterations, producing high quality object boxes at a minor additional computational expense. The architecture achieves impressive gains in performance (up to 6% improvement in detection accuracy) at fast run-time speed (0.22 second per frame on 1242x375 sized images).

Refining Deep Vehicle Detectors for Autonomous Driving

Rakesh Nattoji Rajaram, Eshed Ohn-Bar, and Mohan Manubhai Trivedi
Laboratory for Intelligent and Safe Automobiles
University of California San Diego
{rnattoji, eohnbar, mtrivedi}@ucsd.edu

***Abstract*—Robust and accurate object detection is an essential component of an intelligent vehicle. In particular for on-road applications, localization quality of objects in the image plane is important for accurate distance estimation and consequently safe and smooth trajectory planning. In this paper, we investigate a novel strategy to improve localization accuracy of detected objects, in particular on-road vehicles, using a deep convolutional neural network. An iterative region-of-interest pooling framework is proposed for predicting increasingly tight object boxes. The method is shown to converge within 2-3 iterations, producing high quality object boxes at a minor additional computational expense. The architecture achieves impressive gains in performance (up to 6% improvement in detection accuracy) at fast run-time speed (0.22 second per frame on 1242×375 sized images).**

***Index Terms*—Vehicle detection, localization refinement, convolutional networks, deep learning, fast detection, autonomous driving.**

I. INTRODUCTION

Accurate vehicle localization is an important challenge in the field of intelligent vehicle [1], [2]. In order to understand the dynamics of the surrounding scene, it is essential to continuously detect and accurately localized vehicles [3]. This work proposes the addition of an iterative refinement module to improve object detection and localization performance with a minor increase in computation.

In recent years, techniques reliant on deep convolution neural network (CNN) have been successfully applied to various computer vision problems such as image classification [4], [5], [6], object detection [7], [8], [9], [10], and semantic segmentation [11], [12], [13], [14], thanks to its ability to learn generic features that are robust to intra-class variations and capture context. For object detection, R-CNN [7] has been a common choice among researchers as a state-of-the-art object detector where a CNN classification network is used to independently classify object proposals. This approach is computationally very expensive since the network has to perform a forward pass on each proposal. This technique effectively applies the convolution filters repeatedly at the same location albeit at different scales. Fast R-CNN [8] solves this problem by performing a single pass on the input image and sharing the convolution channel features among proposals. Compared to R-CNN, Fast R-CNN improves the computational efficiency of by an order of magnitude with negligible change in performance. Often, in situations where the object (such as vehicles) appear with large variations in scale, Fast R-CNN [8] is repeatedly applied at multiple scales. The common issue with Fast R-CNN is that its performance

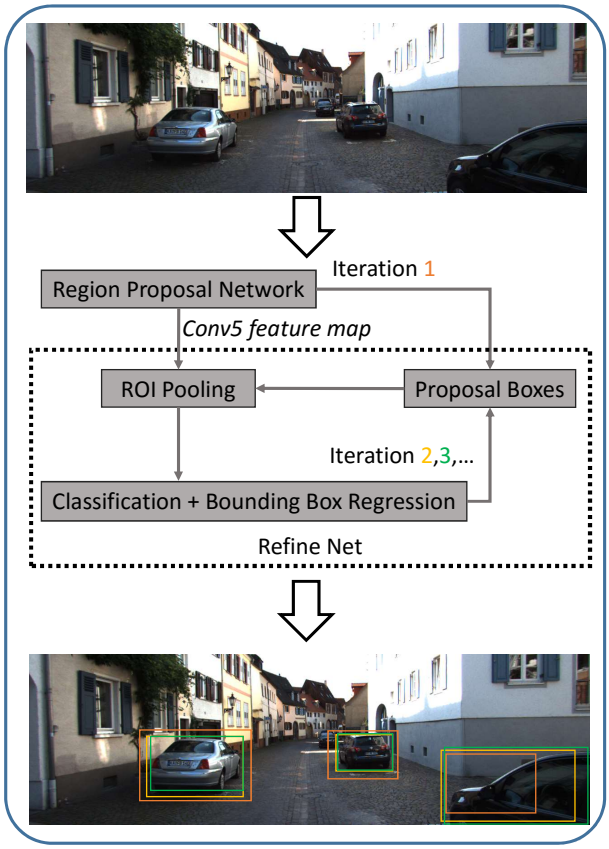


Fig. 1. We present a new strategy to improve the quality of vehicle detection. Our iterative refinement of bounding boxes make use of the already extracted CNN features and improves the localization accuracy of the detection boxes at a marginal increase in computation cost. In the image, orange, yellow and green color represents bounding boxes at iterations 1, 2 and 3 respectively.

relies on the quality region proposals. To solve this problem, Faster R-CNN [9] implements a region proposal network within the CNN framework. This allows for an end-to-end object detection with a single pass through the CNN network along with improving computation efficiency. Despite it's success, these approaches have certain drawbacks. First, Faster R-CNN doesn't handle small objects very well. Currently, this is solved by significantly up-sampling the input image. Together with the use of deep networks, this adds significant computation cost limiting their use in intelligent vehicles. Second, the region of interest (ROI) pooling layer pools features in accordance with proposal boxes locations which are often poorly localized. These features may not accurately

represent the underlying object's characteristics.

In this work, we attempt to address the aforementioned issues by proposing a new strategy for iterative bounding box refinement (see Fig. 1). We call the proposed method RefineNet. The idea is to let the ROI pooling layer pool features from regions closer to the object ground truth by making use of the regressed bounding boxes from the previous iteration. We also perform a quantitative analysis on the impact of various parameters on speed and accuracy and gain useful insights regarding the effect of some parameters on the performance of the detector.

We evaluate our method on KITTI object detection benchmark [15] and show that comparable accuracy can be achieved with RefineNet using a smaller network (ZF Net [16]) as opposed to using Faster R-CNN with a bigger network (VGG16 [5]). Also, RefineNet with ZF runs roughly $9\times$ faster than Faster R-CNN with VGG16 making it one of the fastest detectors on the benchmark.

The contributions presented in this paper are as following,

- **Localization framework**: The insight that better localized ROI leads to better bounding box regression allows the formulation of a novel iterative framework for refinement of object detection boxes. Up to 6% improvement in detection accuracy is observed on the challenging KITTI benchmark [15] and a multi-perspective US highway dataset. Accurate localization is essential for better understanding of surround activities [17], [18] and safe trajectory planning in on-road settings [19].
- **Fast run-time**: The aforementioned insight allows for employing a light-weight network with a small number of region proposals without compromise in detection performance. Furthermore, the proposed refinement modifications are not computationally expensive. Specifically, the entire framework runs at an impressive 0.22 second per frame on 1242×375 sized images, beating in speed all other current state-of-the-art detectors.
- **Evaluations**: The proposed RefineNet network is carefully analyzed for convergence and localization properties on both German/Urban driving settings (KITTI) and a multi-view US highway dataset captured by our lab [18].

II. Related Works

Recent progress in object detection can be attributed to the ability of deep convolutional neural network to learn discriminative features across wide variation in object appearance. In [20], bounding box for object is treated as a regression problem on top of prefixed object masks. R-CNN [7] first minimizes the search space from millions of windows to a few thousand probable windows (using selective search [21]) and then extracts CNN features from each window using a model that is fine tuned on a particular dataset. This high dimensional feature is then passed on to a support vector machine for classification and regression to correct the bounding box. Fast R-CNN [8] builds on top of R-CNN to improve the computational efficiency by introducing ROI pooling layer to extract features by sampling from this layer. Faster R-CNN [9] further improves the computational efficiency by introducing a proposal regression layer to perform object detection with a single pass. 3DOP [22] provides a new proposal generation mechanism using depth information which when used along with Fast-RCNN produces state-of-the art results. Depth can also be used as a cue in the detection model, as in [23]. MSS [24] employs an improved localization model over multi-scale conv5 features. Detection models are learned for each image scale to better capture object variation due to scale. SDP [25] extends the Fast R-CNN idea by introducing ROI pooling layer at multiple *conv* layers to improve detection of small objects and also apply the cascaded rejection classifier technique to quickly reject proposals with low confidence.

Prior to CNN making headway into object detection, Deformable Parts Model (DPM) [26] was the gold standard for years. The key idea introduced in DPM was to formulate an object as a root template with a fixed number of associated parts whose position is flexible relative to the root template. Similarly, Regionlets [27] introduces appearance flexibility but in the feature space. It operates by minimizing the search space to a few thousand windows (using selective search [21]), extracting features from a fixed number of regions inside these windows, and then pooling them to establish invariance to localization, scale and aspect ratio. Next, the detected objects are re-localized using a localization model. SubCat [28] introduces modifications on top of the detector. Here, objects are sub-categorized into a fixed number of clusters based on geometric features such as height, width, aspect ratio, occlusion etc. and other image features. Then, a separate model is trained for each of these clusters. Along with improving detector accuracy, SubCat also estimated vehicle orientation.

It is possible to draw some similarities between our work in this paper to recurrent neural network (RNN) or auto-context work such as [29], [30], [31]. We note that the aforementioned approaches do not iterate pooling of the CNN features in each ROI as proposed in this work and do not study such as framework for improved object localization for on-road data. Hence, the architecture proposed in this paper and the analysis of its improvement and convergence properties have not been presented in related studies.

III. RefineNet

In this section, we introduce our strategy to iteratively refine bounding box locations to improve localization accuracy. We call this approach as RefineNet. Fig. 2 provides the overall architecture of RefineNet. It is built on top of Faster R-CNN [9] object detector. Before we go into the details of RefineNet, we briefly cover relevant parts of Fast R-CNN and Faster R-CNN.

A. Fast R-CNN

Fast R-CNN [8] improves upon R-CNN [7] by approximating features using an region of interest (ROI) pooling layer. Let I be an input image of spatial dimensions $H \times W$. First, convolutional feature maps are extracted over the input image. With ZF [16] network, this is the output corresponding to *conv5*

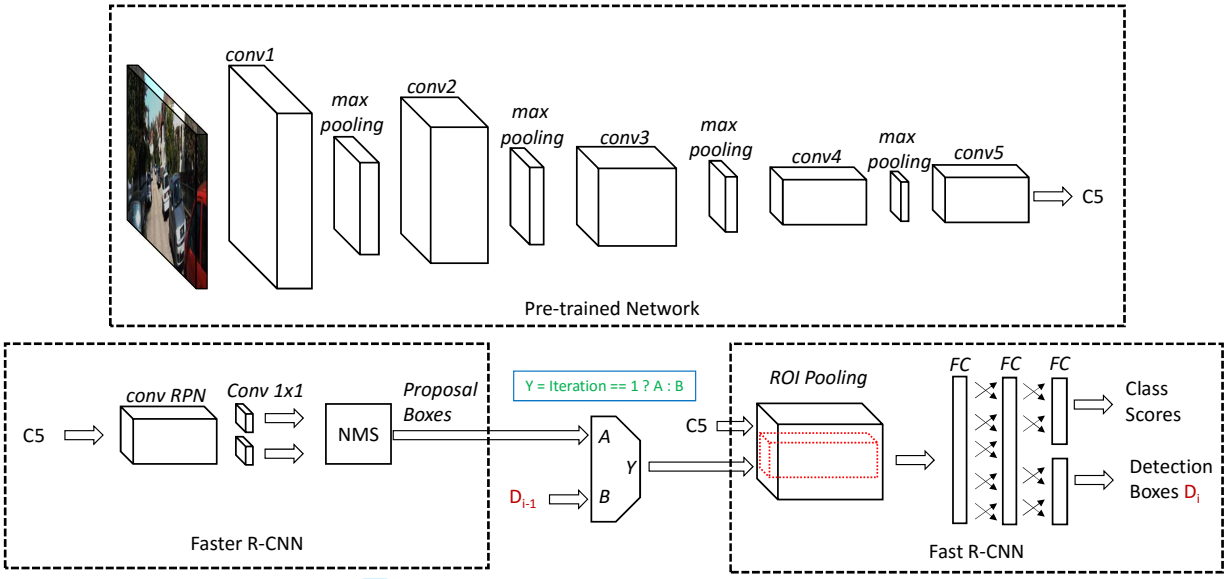


Fig. 2. RefineNet: A pretrained network which is fine-tuned on a object detection dataset is used to extract convolutional feature map (C_5). Using these features, proposal boxes are generated with Faster R-CNN framework. Again with C_5 features and proposal boxes, detection boxes D_1 are generated with Fast R-CNN framework .This constitutes the first iteration. Successive iterations i involves refining the detection boxes D_i by using detections from previous iterations i.e. D_{i-1} as proposal boxes for constructing the ROI pooled features.

layer and is a matrix (C_5) of dimensions $\lfloor \frac{H}{16} \rfloor \times \lfloor \frac{W}{16} \rfloor \times 256$. Here 16 represents the factor by which the input image gets scaled after passing through the *conv5* layer. This is also true for AlexNet [4] and VGG16 [5]. From C_5, features corresponding to each proposal box is extracted by pooling features from the corresponding spatial location and re-sampling them to a fixed size ($6 \times 6 \times 256$ for ZF network). Bounding box regression and class score are modeled by mapping the pooled features using 3 cascaded fully connected layers.

B. Faster R-CNN

Faster R-CNN [9] is built upon two modules namely the region proposal network (RPN) and Fast R-CNN [8]. The entire system is an single pass end-to-end unified network for object detection. The RPN layer replaces the proposal boxes input that was fed into the Fast R-CNN framework.

The region proposal network takes an image as input and outputs proposal boxes along with objectness score. This process is modeled as a regression problem over the convolutional feature map that is extracted from the input image (C_5 is used to share computational cost with Fast R-CNN). To generate region proposals, a small convolutional filter of size 3×3 spatial window is applied over the input features followed by two 1×1 convolutional filters for generating proposal boxes and objectness scores at each spatial location. At each spatial location, multiple proposals can be generated using *anchor* boxes. These anchor boxes can be setup at multiple scales and aspect ratio and serves as reference for regression i.e. if the k^{th} anchor box is defined as $a_k = \{x_k, y_k, w_k, h_k\}$ then the regression targets would be $\{\delta x_k, \delta y_k, \delta w_k, \delta h_k\}$ where $\delta x_k = x_k - x_g$ and so on. Here the subscript g corresponds to the closest ground truth box. This formulation allows proposal generation at multiple scales and aspect ratio without constructing feature pyramid.

C. RefineNet - Formulation

In supervised CNN frameworks, the objective is to train a network F that predicts an output $\hat{y}$, given an input x (i.e. an image),

$$\hat{y} = F(\mathbf{x}) \tag{1}$$

In this notation, F is an embedding of all of the parameters and operations of the network layers (shown in Fig. 2). A key insight in the R-CNN-based frameworks is the additional ROI parameters, D, so that the prediction function becomes

$$\hat{y} = F(\mathbf{x}, D) \tag{2}$$

We note that the label space is now both a class prediction and a $4D$ bounding box, $\hat{y} = \{\hat{y}^c, \hat{y}^D\}$. The parameter $D \in R^{M \times 4}$ specifies M boxes in the image plane and is introduced for detection and localization applications (as opposed to the classification only case in Eqn. 1). Although current state-of-the-art object detectors all employ a region proposal and pooling mechanism, several potential questions have not been well studied in literature, in particular the impact of a poorly localized D on the output of the fully connected layers and output quality. Furthermore, y^D is obtained using a bounding-box regression module, and its ability to recover from poorly localized regions D also requires analysis.

Motivated by such potential issues, we introduce a generalization of R-CNN with an iterative framework. Since $\hat{y}^D$ has undergone bounding-box regression, y^D is generally better localized then the input proposal ROIs, D. We then re-feed

y^D to analyze for further gains, and define $\hat{y}_2 = F(\mathbf{x}, y^D) := F(\mathbf{x}, D_1)$. In general, the process can be applied iteratively,

$$\hat{y}_{N+1} = F(\mathbf{x}, D_N) \tag{3}$$

where we note that the N=1 case are existing R-CNN techniques. Throughout the iterations, the D parameter changes from the RPN output in $N = 1$, and previous R-CNN regression outputs at $N > 1$, until the regression module provides no additional refinement benefit. This formulation allows us to analyze the properties of the bounding box regression module in F.

D. RefineNet Training

Training the RefineNet is similar to training the Faster R-CNN network. It follows the 4 step alternate training.

1) Train the RPN network initialized with model pre-trained on ImageNet [32] dataset.
2) Train the Fast R-CNN network (from random initialization) using proposals generated in step 1.
3) Fix convolution layers and fine-tune RPN network from step 2.
4) Fix convolution layers and fine-tune Fast R-CNN network from step 3.

For training the RPN network, each anchor box is assigned a class label and regression target at each location. Labels are assigned in the following order. (i) If an anchor box has intersection over area of don't care box overlap greater than 0.6, then the anchor box is ignored. (ii) If an anchor box has intersection over union (IoU) overlap greater than 0.5, then the anchor box is considered foreground. (iii) For all the ground truth boxes that are not assigned any anchor box, the anchor box is with highest IoU is used. (iv) All the anchor boxes that have IoU overlap with all the ground truth boxes less than 0.3 are considered as background.

The loss function used for training the RefineNet follows from Faster R-CNN where a multi-task loss function is defined to learn both classification and regression together.

E. RefineNet Testing

Testing the RefineNet is an iterative process. In the first iteration detection boxes D_i are generated using Faster R-CNN framework. We store the already computed C_5 convolutional feature map in memory. Successive iterations i uses C_5 and detection boxes from previous iterations i.e D_{i-1} as proposal boxes input in the ROI pooling stage of Fast R-CNN. At every iteration, the detection boxes achieves higher overlap with the ground truth boxes. Therefore, the features pooled in the successive iterations will better represent the underlying object class and it's location. This allows for recursively improving the classification score and also the localization accuracy.

IV. Experiments

We evaluate our method on KITTI object detection benchmark [15]. The dataset is split into training and validation as suggested in [33]. The training and validation set have 3682 and 3799 images respectively. We augment the training set by including horizontally flipped version of the images. KITTI object detection benchmark evaluates the detector performance at 3 different difficulty settings differentiated based on constraints in Table I. We train and test on moderate difficult settings, as suggested by the KITTI benchmark [34]. Hard ground truth boxes are considered as don't care boxes during both training and testing. We evaluate area under the Precision-Recall curve (AUC) as a measure of detector's performance. A detection box is considered as true positive if any ground truth has an IoU overlap greater than o_{th}.

TABLE I
Distribution of vehicles in KITTI dataset into different difficulty setting based on height, occlusion and truncation.

Difficulty	Height	Occlusion	Truncation
Easy	40	Fully Visible	15%
Moderate	25	Partially Occluded	30%
Hard	25	Difficult to See	50%

We train and test region proposal and object detection network on images at multiple scales. This has been the trend in CNN based object detection on KITTI object detection benchmark [15]. For example, in [35], input image is up-sampled by $4\times$ and in [22] by $3\times$. This could be due the following reasons. (i) Convolution layers of CNN has stride greater than 1. (ii) Max-pooling layers reduce spatial dimensions. (iii) Because, the CNN network is pre-trained at a fixed scale of 224×224, it is unable to generate rich features for objects at different scales. We train and test at different combinations of scales such that the shortest side has s pixels.

During training, each ground truth is assigned to the closest scale. During testing, only the top K_2 proposals are selected after passing the top K_1 proposals through a non maximal suppression (NMS) unit (IoU threshold: 0.7). Also, detection occurs at each scale independently which are later concatenated and passed through a NMS unit (IoU threshold: 0.3) to remove duplicate detection boxes.

The following parameters are used for the 4-step alternate training using stochastic gradient descent with momentum. (i & iii) Batch size: 256. Total iterations: 80,000. Base learning rate: 0.001. Step size: 60,000. Learning rate scale factor: 0.1. Momentum: 0.9. Weight decay: 0.0001. (ii & iv) Batch size: 128. Total iterations: 40,000. Rest of the parameters remain unchanged.

We train a RefineNet model (M_1) using ZF network on the KITTI training split. Training and testing is carried out at scales $s = \{375, 750\}$ which is $\{1\times, 2\times\}$ the image size respectively. We use the default set of 9 anchors at 3 different scales (8,16 and 32) and 3 different aspect ratios (1:1, 1:2 and 2:1). For iteration 1, we use the $K_1 = 6000$ and $K_2 = 300$ boxes into the Fast R-CNN network. In Table II, we report

[1]Using Nvidia GTX Titan X

TABLE II
AUC AT DIFFERENT OVERLAP THRESHOLD (o_{th}) AND NUMBER OF REFINEMENT ITERATIONS (N). METRICS GENERATED ON KITTI VALIDATION SET USING THE REFINENET MODEL M_1.

o_{th}	N=1	N=2	N=3	N=4	N=5
0.60	90.92	**91.97**	91.52	91.30	91.23
0.65	86.48	**88.34**	87.76	87.67	87.61
0.70	78.78	81.26	**81.58**	81.15	80.73
0.75	65.10	**69.63**	69.28	69.59	69.20
0.80	43.63	50.06	**51.61**	51.21	50.86
Runtime[1] (sec)	0.20	0.24	0.29	0.34	0.38

TABLE III
AUC AT DIFFERENT NUMBER OF INPUT PROPOSALS (K_2) AND NUMBER OF REFINEMENT ITERATIONS (N). METRICS GENERATED ON KITTI VALIDATION SET USING THE REFINENET MODEL M_1.

K_2	N=1	N=2	N=3	N=4	N=5	Runtime(sec) @N=3
200	78.79	81.07	**81.25**	80.86	80.54	0.22
100	78.83	81.13	81.15	81.02	80.44	0.20
50	78.16	80.66	80.79	80.46	80.00	0.16
25	76.77	78.99	79.06	78.97	78.35	0.15

TABLE IV
AUC AT DIFFERENT NUMBER OF INPUT PROPOSALS (K_2) AND NUMBER OF REFINEMENT ITERATIONS (N). METRICS GENERATED ON KITTI VALIDATION SET USING THE REFINENET MODEL M_2.

K_2	N=1	N=2	N=3	N=4	N=5	Runtime(sec) @N=3
200	74.54	80.03	**80.69**	80.13	78.83	0.20
100	74.79	80.02	80.37	79.41	78.28	0.18

AUC as a function of o_{th} and number of refinement iterations. As the o_{th} increases, the improvement in AUC increases from 1.05% to 7.98%. Clearly as the overlap requirement becomes stricter, the refinement step shows greater improvement and demonstrates it's ability to improve localization accuracy with just 1 or 2 iterations. At $o_{th} = 0.7$, with $N = 3$, M_1 achieves maximum AUC of **81.58**%.

In Table III, we report accuracy and runtime as a function of K_2. We fix $K_1 = 1000$ and $o_{th} = 0.7$. We report the runtime at $N = 3$ (i.e. improvement stops after $N = 3$). At $K_2 = 200$, runtime reduces from 0.29 seconds to **0.22** seconds with less than 0.4% decrease in AUC.

In the second experiment, we study the effect of number of anchor boxes. Specifically, we train a RefineNet model (M_2) with just one anchor box (a square with sides of length 67 pixels and centered at 0,0). Again, training and testing is carried out at scales $s = \{375, 750\}$. In Table IV, we report accuracy and runtime as a function of K_2. Guided by our previous experiment, we set $K_1 = 1000$. At $K_2 = 200$, runtime reduces to **0.20** seconds with less than 0.9% decrease in AUC. Although, the decrease in runtime is not significant, the improvement in AUC from 74.54% to 80.69% is more than **6%**. By decreasing the number of anchor boxes from 9 in M_1 to just 1 in M_2 we have reduced the number of model parameters. Intuitively, this led to 4% decrease in AUC (78.79% vs 74.54%) at the first iteration but, RefineNet was able to improve the quality of detection in just 2 additional iterations.

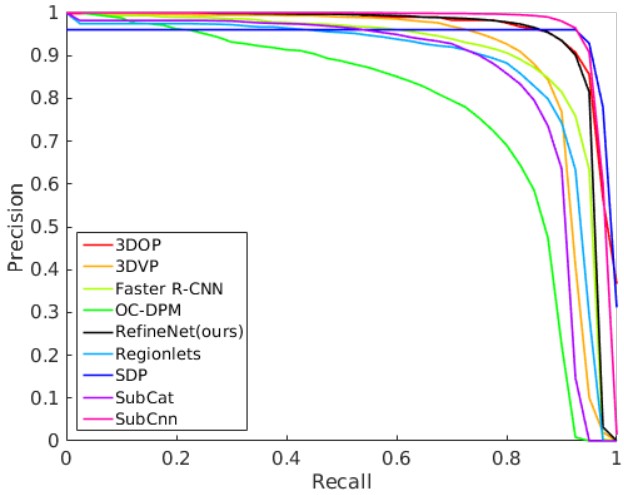


Fig. 3. Precision-Recall curves: Comparison between state-of-the-art vehicle detectors on KITTI object detection benchmark at easy difficulty settings.

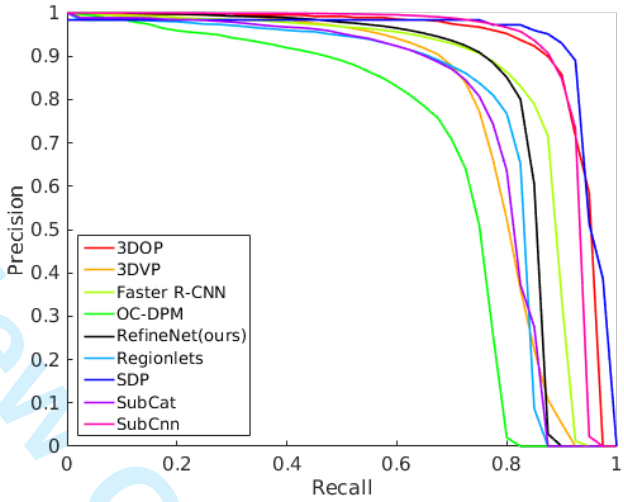


Fig. 4. Precision-Recall curves: Comparison between state-of-the-art vehicle detectors on KITTI object detection benchmark at moderate difficulty settings.

Comparing VGG16 and ZF: Most of the experiments employ the ZF Net [16]) which allows fast training and testing, but the larger VGG16 [5] network is also commonly employed. With the VGG16 network, the same settings as the experiments in Table I, and an overlap threshold of $o_{th} = 0.70$, the AUC for $N = 1, 2, 3, 4$ is $82.20, 83.87, 83.27, 83.35$, respectively. An improvement in the initial iterations is observed, as with the ZF network, after which little benefit is shown by further refinement. Furthermore, the ZF network achieves with refinement only 2.29 accuracy points below the detection performance of the larger VGG16 network but with a run-time of nearly an order of magnitude greater.

Evaluation on KITTI test set: We train a RefineNet model

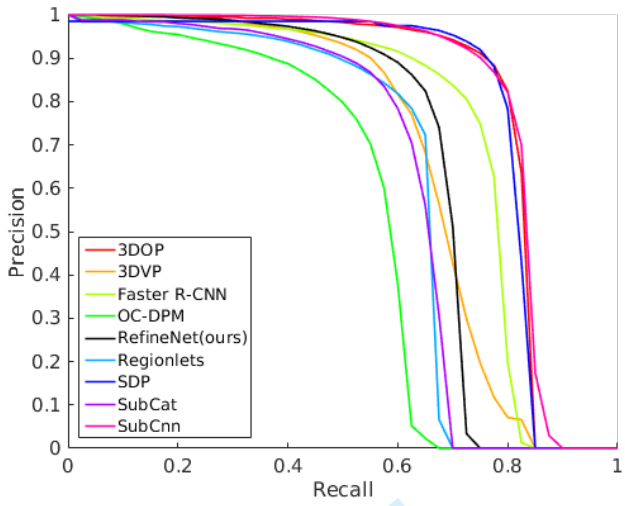


Fig. 5. Precision-Recall curves: Comparison between state-of-the-art vehicle detectors on KITTI object detection benchmark at hard difficulty settings.

TABLE V
AUC ACHIEVED BY STATE-OF-THE-ART DETECTORS ON KITTI OBJECT DETECTION BENCHMARK AT MODERATE DIFFICULTY SETTINGS. ASTERISK (*)- METHODS EMPLOY THE VGG [5] NETWORK AS OPPOSED TO THE ZF NETWORK USED IN THIS WORK.

Detector	AUC			Runtime(sec)
	Moderate	Easy	Hard	
SubCNN* [35]	**89.04**	90.81	**79.27**	2
SDP* [25]	88.85	90.14	78.38	0.40
3DOP* [22]	88.64	**93.04**	79.10	3
Faster R-CNN* [9]	81.84	86.71	71.12	2
RefineNet (ours)	79.17	89.88	66.38	**0.22**
Regionlets [27]	76.45	84.75	59.70	1
SubCat [28]	75.46	84.14	59.71	0.7
3DVP [33]	75.77	87.46	65.38	40
OC-DPM [36]	65.95	74.94	53.86	10

with parameters taken from M_1 and on the entire training set and submit the results to the online evaluation server of KITTI. This model achieves 89.88%, 79.17%, and 66.38% on the easy, moderate, and hard settings KITTI benchmark [15], respectively. Fig. 3 compares PR curves with the state-of-the art techniques. In Table V, we compare AUC at different different difficulty settings. First, we note that all state-of-the-art detectors (including Faster R-CNN) employ the more powerful but more computationally expensive VGG16 [5] network. On the other hand, our model is light weight in memory and run-time, yet reaching state-of-the-art on 'easy' settings. Furthermore, the RefineNet model performs nearly at the same performance level on moderate settings as the Faster R-CNN model which serves as the closest baseline (not employing other complementary modifications as in SubCNN [35] and 3DOP [22]). As moderate settings include challenging cases of higher truncation, occlusion, and small sized objects, the results are encouraging as RefineNet is an order of magnitude faster in run-time, and is also significantly faster to train.

Fig. 6 demonstrates the improvement due to the iterative refinement on a variety of scenes and visual challenges. In particular, large pose variation, occlusion, truncated, and small instances are all shown to be better handled by RefineNet compared to the baseline. The figure demonstrates how RefineNet better captures the true geometry of objects, seen by tighter boxes and correct object boundary identification, even when an object is occluded by another object. The improved localization is crucial for driver assistance applications, where 3D distance is often estimated using the object location and size in the scene.

Fig. 7 demonstrates cases which still need to be resolved in future work. These include severe occlusion by other objects, where RefineNet may improve but not fully recover the correct location of an object. Truncation is also shown to challenging due to the lack of information to operate on in the detection and localization process. Further modifications to the proposed framework are needed to handle such challenging cases.

Evaluation on US highway settings: In order to study generalization of the proposed framework, we test RefineNet with a different dataset of US highway settings collected using a four perspective setup [18]. The capture is done using four synchronized GoPro cameras, designed for panoramic analysis of surround vehicles and their behavior. The dataset contains 400 frames in each view (a total of 1600 frames) and over 4000 vehicles. Training is performed on KITTI, which is front-view only but contains vehicles at all orientations relative to the ego-vehicle due to the presence of many intersections. For evaluation, we follow KITTI with a 70% overlap threshold and evaluation on three truncation and occlusion levels (shown in Fig. 9. A significant improvement in detection performance due to the proposed approach is shown in all perspectives and evaluation settings. As shown in Fig. 8, perspectives with similar views to KITTI (front and rear) particularly benefit the refinement with RefineNet, by up to $5-6\%$ AUC increase. Side views on the other hand contain appearance variations leading to aspect ratios which are not found in KITTI. Consequently, although overall performance increase with RefineNet, the model often improves localization in one dimension of the bounding box while reducing localization in another. Generally, as side views often contain distortion due to the perspective of the camera, further studies are required for improving generalization over such scenes. Example cases are shown in Figs. 10 and 11, showing improvement due to refinement on both large and small object instances.

V. CONCLUDING REMARKS

In this paper, we introduce a new strategy called RefineNet to improve localization accuracy of vehicle detection and report a gain of up to **6%** in AUC. Our method is efficient as it relies on using already computed features in each iteration and provides top performance with fast detection. Specifically, RefineNet runs in about **0.22** seconds per image on 1242×375 size images, allowing for smaller convolutional neural networks to operate on similar same performance level to very deep and large networks. On KITTI object detection benchmark, it achieve 79.19% on moderate difficulty settings.

Fig. 6. Improvement of RefineNet is shown under occlusion, pose variation, and variation in size. Sample detection boxes generated using RefineNet model M_2 on KITTI validation set. In the image, orange, yellow and green color represents bounding boxes at iterations 1, 2 and 3 respectively. Confidence scores are represented as numbers adjacent to the detection boxes.

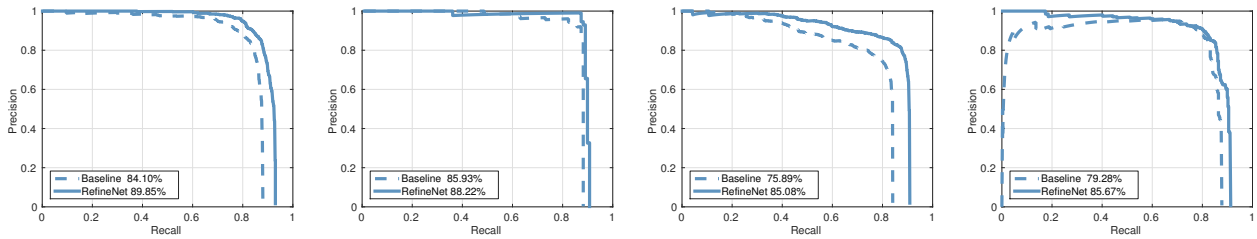
Fig. 7. Challenging cases for RefineNet. Truncation and occlusion cases are improved but not resolved due to the iterative refinement. Sample detection boxes generated using RefineNet model M_2 on KITTI validation set. In the image, orange, yellow and green color represents bounding boxes at iterations 1, 2 and 3 respectively. Confidence scores are represented as numbers adjacent to the detection boxes.

It is the fastest detector that achieves upwards of 70% AUC. On easy difficulty settings, it achieves 89.88% AUC which is close to state-of-the-art result.

In the future, utilization of scene information [12], [37] in generating or pruning proposals can further provide increased run-time speed without sacrificing detection and localization quality. The results suggest large performance gains on localization of objects with little occlusion, little truncation, and large size. A multi-resolution analysis [38], [39], [40] can further benefit detection of small instances. Data augmentation [41], [42] can benefit the trained model's ability to handle occlusion and truncated instances.

REFERENCES

[1] S. Sivaraman and M. M. Trivedi, "Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking and behavior analysis," *IEEE Transactions on Intelligent Transportation Systems*, 2013.



(a) Front (b) Right (c) Rear (d) Left

Fig. 8. Performance curves with and without the proposed refinement on the multi-perspective highway dataset. Area under the curve improvement is shown for each of the perspectives. Evaluation includes partial occlusion and partial truncation instances.

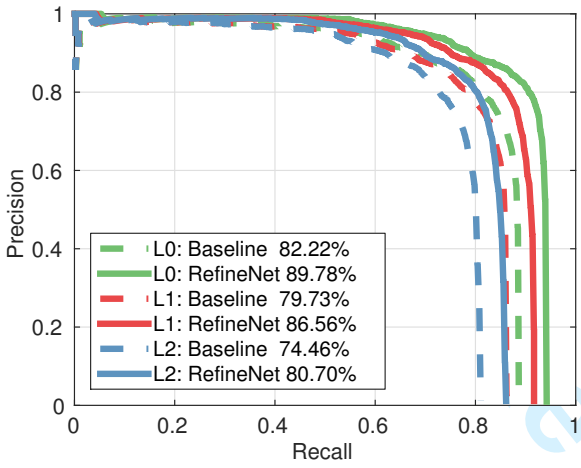


Fig. 9. Improvement due to the proposed refinement framework for different evaluation settings of 'L0' - no occlusion or truncation, 'L1' - partial occlusion and truncation, 'L2' - all instances, including heavy occlusion and truncation. The precision-recall curves for each evaluation setting are computed over all of the four perspectives.

[2] S. Sivaraman, B. Morris, and M. M. Trivedi, "Observing on-road vehicle behavior: Issues, approaches, and perspectives," in *IEEE Conf. Intelligent Transportation Systems*, 2013.

[3] B. Ranft and C. Stiller, "The role of machine vision for intelligent vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 8–19, 2016.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., 2012, pp. 1097–1105.

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[6] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014.

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Computer Vision and Pattern Recognition*, 2014.

[8] R. Girshick, "Fast r-cnn," in *International Conference on Computer Vision*, 2015.

[9] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, 2015.

[10] L. Huang, Y. Yang, Y. Deng, and Y. Yu, "Densebox: Unifying landmark localization with end to end object detection," *CoRR*, vol. abs/1509.04874, 2015.

[11] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[12] E. Romera, L. M. Bergasa, and R. Arroyo, "Can we unify monocular detectors for autonomous driving by using the pixel-wise semantic segmentation of cnns?" in *IEEE Intelligent Vehicles Symposium*, 2016.

[13] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *CoRR*, vol. abs/1511.07122, 2015.

[14] G. Lin, C. Shen, I. D. Reid, and A. van den Hengel, "Efficient piecewise training of deep structured models for semantic segmentation," *CoRR*, vol. abs/1504.01013, 2015.

[15] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 3354–3361.

[16] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *CoRR*, vol. abs/1311.2901, 2013.

[17] A. Doshi and M. M. Trivedi, "Tactical driver behavior prediction and intent inference: A review," in *IEEE Conf. Intelligent Transportation Systems*, 2011.

[18] J. Dueholm, M. Kristoffersen, R. Satszoda, T. Moeslund, and M. M. Trivedi, "Trajectories and behaviors of surrounding vehicles using panoramic camera arrays," *IEEE Transactions on Intelligent Vehicles*, 2016.

[19] E. Ohn-Bar and M. M. Trivedi, "Looking at humans in the age of self-driving and highly automated vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 90–104, 2016.

[20] C. Szegedy, A. Toshev, and D. Erhan, "Deep neural networks for object detection," in *Advances in Neural Information Processing Systems*, 2013, pp. 2553–2561.

[21] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[22] X. Chen, K. Kundu, Y. Zhu, A. Berneshawi, H. Ma, S. Fidler, and R. Urtasun, "3d object proposals for accurate object class detection," in *Advances in Neural Information Processing Systems*, 2015.

[23] J. J. Yebes, L. M. Bergasa, and M. García-Garrido, "Visual object recognition with 3d-aware features in kitti urban scenes," *Sensors*, vol. 15, no. 4, pp. 9228–9250, 2015.

[24] E. Ohn-Bar and M. M. Trivedi, "Multi-scale volumes for deep object detection and localization," *Pattern Recognition*, 2016.

[25] F. Yang, W. Choi, and Y. Lin, "Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers," in *Conference on Computer Vision and Pattern Recognition*, 2016.

[26] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[27] X. Wang, M. Yang, S. Zhu, and Y. Lin, "Regionlets for generic object detection," in *International Conference on Computer Vision*, 2013.

[28] E. Ohn-Bar and M. M. Trivedi, "Learning to detect vehicles by clustering appearance patterns," *IEEE Transactions on Intelligent Transportation Systems*, 2015.

[29] Z. Tu, "Auto-context and its application to high-level vision tasks," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[30] J. Ba, V. Mnih, and K. Kavukcuoglu, "Multiple object recognition with visual attention," *CoRR*, vol. abs/1412.7755, 2014.

[31] K. Gregor, I. Danihelka, A. Graves, and D. Wierstra, "DRAW: A recurrent neural network for image generation," *CoRR*, vol. abs/1502.04623, 2015.

(a)

(b)

Fig. 10. RefineNet results on a four-perspective US highway dataset with training on KITTI. In general, vehicles in the front view are better detected with RefineNet over the baseline, as shown in (a) and (b), while side view vehicles offer challenges due to variations in appearance which is not found in KITTI.

[32] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[33] Y. Xiang, W. Choi, Y. Lin, and S. Savarese, "Data-driven 3d voxel patterns for object category recognition," in *Computer Vision and Pattern Recognition*, 2015.

[34] J. J. Yebes, L. M. Bergasa, R. Arroyo, and A. Lázaro, "Supervised learning and evaluation of kittis cars detector with dpm," in *IEEE Intelligent Vehicles Symposium*, 2014.

[35] Y. Xiang, W. Choi, Y. Lin, and S. Savarese, "Subcategory-aware convolutional neural networks for object proposals and detection," in *arXiv:1604.04693*, 2016.

[36] B. Pepik, M. Stark, P. Gehler, and B. Schiele, "Occlusion patterns for object class detection," in *Conference on Computer Vision and Pattern*

(a)

(b)

Fig. 11. RefineNet results on a four-perspective US highway dataset with training on KITTI.

Recognition, 2013.

[37] A. D. Costea and S. Nedevschi, "Multi-class segmentation for traffic scenarios at over 50 fps," in *IEEE Intelligent Vehicles Symposium*, 2014.

[38] R. N. Rajaram, E. Ohn-Bar, and M. M. Trivedi, "Looking at pedestrians at different scales: A multiresolution approach and evaluations," *IEEE Transactions on Intelligent Transportation Systems*, 2016.

[39] A. D. Costea, A. V. Vesa, and S. Nedevschi, "Fast pedestrian detection for mobile devices," in *IEEE Conference on Intelligent Transportation Systems*, 2015.

[40] R. N. Rajaram, E. Ohn-Bar, and M. M. Trivedi, "An exploration of why and when pedestrian detection fails," in *IEEE Conf. Intelligent Transportation Systems*, 2015.

[41] J. Xu, D. Vázquez, A. M. López, J. Marin, and D. Ponsa, "Learning a part-based pedestrian detector in a virtual world," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 5, pp. 2121–2131, 2014.

[42] J. Xu, D. Vázquez, A. M. López, J. Marin, and D. Ponsa, "Learning a multiview part-based model in virtual world for pedestrian detection," in *IEEE Intelligent Vehicles Symposium*, 2013.