2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)
Windsor Oceanico Hotel, Rio de Janeiro, Brazil, November 1-4, 2016

RefineNet: Iterative Refinement for Accurate Object Localization

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

***Abstract*— We investigate a new strategy for improving localization accuracy of detected vehicles using a deep convolutional neural network. Specifically, we implement an iterative bounding box refinement on top of a state-of-the-art object detector. The bounding box refinement is achieved by iteratively pooling features from previous object location predictions. On KITTI vehicle detection benchmark, we achieve up to 6% improvement in average precision over the baseline results. Furthermore, the proposed refinement framework is computationally light, allowing for object detection at high run-time speeds. Our method runs at $\sim$0.22 seconds per image on images of size 1242×375, making it one of the fastest detectors reported on the KITTI object detection benchmark.**

I. Introduction

Accurate object localization is an important challenge when developing vision-based systems for intelligent vehicles and robots [1]. Essential components for autonomous driving, such as accurate 3D localization of surround objects, surround agent behavior analysis, navigation and planning, and other higher-level vision tasks [2] are all impacted by the quality of the initial object localization. This work studies improving a deep convolution neural network (CNN) object detector by adding a module which iteratively refines object box proposals. The proposed refinement module is light weight, and results in both fast and high-quality detection of objects. We refer to this novel strategy as RefineNet, and analyze its behavior and convergence properties.

In recent years, techniques reliant on deep CNNs have been successfully applied to various computer vision problems such as image classification [3], [4], [5], object detection [6], [7], [8], [9], semantic segmentation [10], [11], [12], and many more, thanks to its ability to learn robust, hierarchical, generic features, applicable to varying tasks.

Region-based CNN [6] (R-CNN) has been proposed for object detection, 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. Fast R-CNN [7] 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 [7] is repeatedly applied at multiple scales. The common issue with Fast R-CNN is that its performance relys on the quality region proposals. To solve this problem, Faster R-CNN [8] 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. Hence, Faster R-CNN is our starting point baseline. 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.

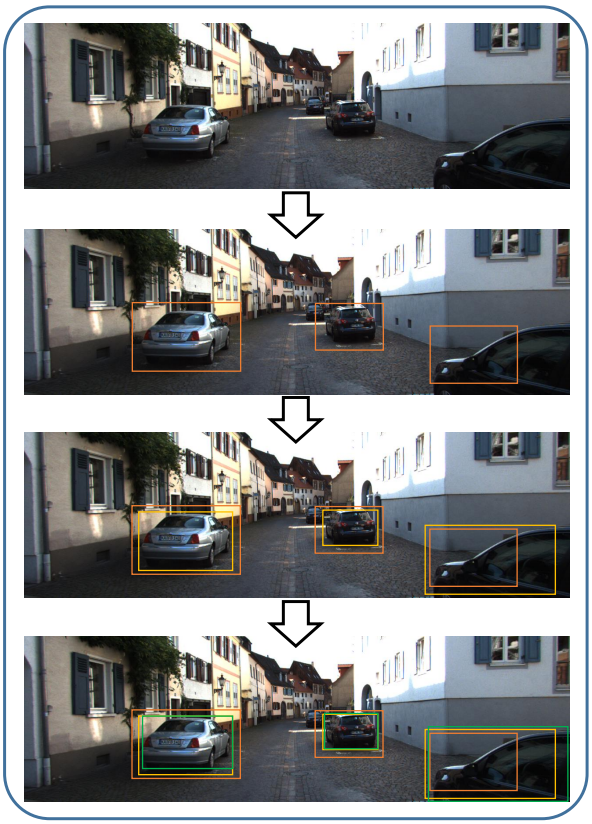

Fig. 1. We present a new strategy (termed RefineNet) to improve vehicle detection and localization accuracy at a marginal increase in computation cost. In the image, orange, yellow and green color represents bounding boxes at iterations 1, 2 and 3 respectively.

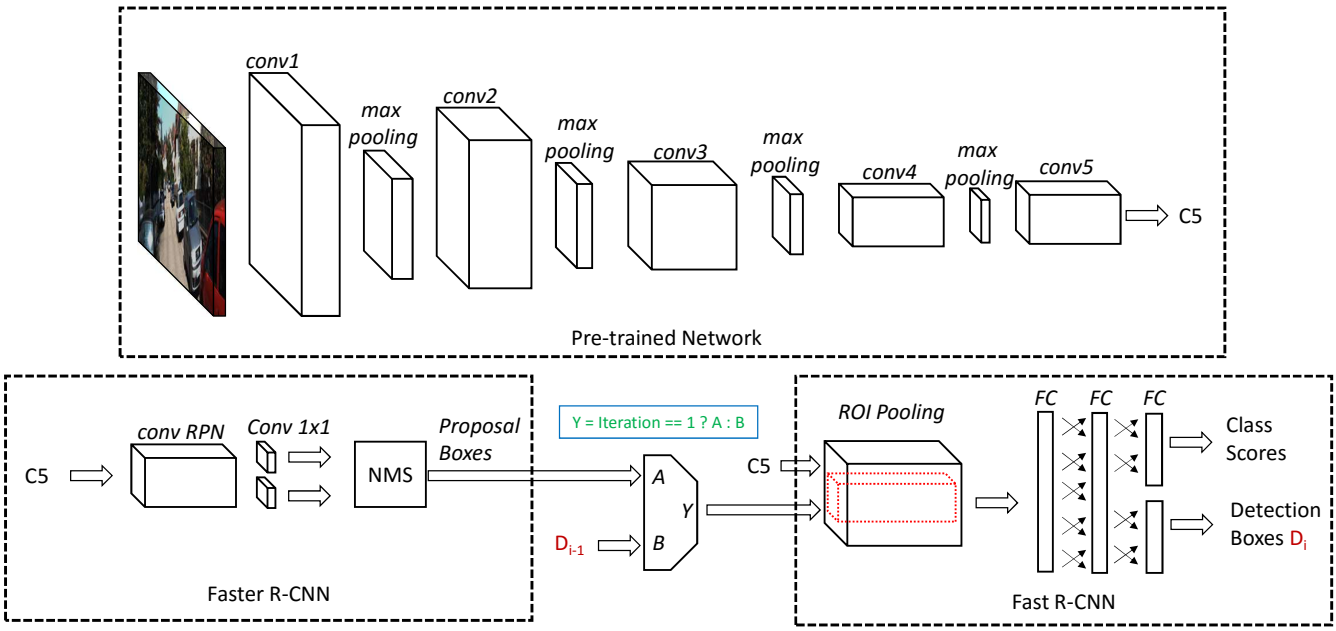

Fig. 2. RefineNet: A pretrained network which is fine-tuned on a object detection dataset is used to extract convolutional feature map (C_5). Using these features, proposal boxes are generated with the Faster R-CNN framework [8]. Next, the operation is repeated with C_5 features and the proposal boxes, so that a refined set of detection boxes D_1 are generated. This constitutes the first iteration in RefineNet. Successive iterations i involves refining the detection boxes D_i by using detections from previous iterations, $D_{i\ \ 1}$, as proposal boxes for constructing the RoI pooled features.

In this work, we attempt to address the aforementioned issues by proposing a new strategy for iterative bounding box refinement (see Fig. 1), called 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 [13] and show that comparable accuracy can be achieved with RefineNet using a smaller network (ZF Net [14]) as opposed to using Faster R-CNN with a bigger network (VGG16 [4]). 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 with little compromise to detection performance.

II. RefineNet

In this section, we introduce our strategy to iteratively refine bounding box locations to improve localization accuracy. Fig. 2 provides the overall architecture of RefineNet. It is built on top of Faster R-CNN [8] 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 [7] improves upon R-CNN [6] 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 [14] network, this is the output corresponding to *conv5* 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 also holds for the AlexNet [3] and VGG16 [4] network architectures. 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 [8] is built upon two modules namely the region proposal network (RPN) and Fast R-CNN [7]. 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 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 [15] 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 an intersection over union (IoU) overlap with a don't care box greater than 0.6, then the anchor box is ignored. (ii) If an anchor box has an 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 (two sibling output layers).

D. RefineNet Testing

Testing the RefineNet is an iterative process. In the first iteration, detection boxes D_i are generated using the Faster R-CNN framework. We store the already computed C_5 convolutional feature map in memory. Next, each successive iterations i use the C_5 and detection boxes from previous iterations, $D_{i\ 1}$, 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 its location. This allows for recursively improving the classification score and also the localization accuracy.

III. Implementation Details

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 [13]. For example, in [16], input image is upsampled by $4\times$ and in [17] 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.

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%

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.

IV. Experiments

We evaluate our method on KITTI object detection benchmark [13] for a car detection task. The dataset is split into training and validation as suggested in [18]. 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. 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}, which is fixed as 0.7 in the experiments.

We train a RefineNet model (M_1) using the 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 (defined in Section III) into the Fast R-CNN network. In Table II, we report AUC as a function of the number of refinement iterations with an overlap of 0.7. At this strict overlap requirement, we demonstrate the ability of the refinement step to improve localization accuracy with just 1 or 2 iterations. With $N = 3$, M_1 achieves maximum AUC of **81.58**%. At $K_2 = 200$, runtime

TABLE II

AUC AT OVERLAP 0.7 VS. THE NUMBER OF REFINEMENT ITERATIONS (N). METRICS GENERATED ON KITTI VALIDATION SET USING THE REFINENET MODEL M_1.

	N=1	N=2	N=3	N=4	N=5
AUC	78.78	81.26	**81.58**	81.15	80.73
Runtime[1] (sec)	0.20	0.24	0.29	0.34	0.38

reduces from 0.29 seconds to **0.22** seconds with less than 0.4% decrease in AUC. As a final 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 $s = \{375, 750\}$. 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 results in a 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.

Evaluation on KITTI object detection benchmark: We train a RefineNet model with parameters taken from M_1 and train it on the entire training set. This model achieves **79.17%** on the KITTI benchmark [13]. In Table III, we compare AUC at different different difficulty settings. SubCNN [16] and 3DOP [17] share maximum AUC at different difficulty settings. However, this comes at significant increase in computation cost. SubCNN employs VGG16 [3] on upsampled input images of up to 4x whereas 3DOP employs VGG16 [4] with upsampled input images by 3.5x. SDP addresses the issue of detecting small objects by using cascaded classifier at different *conv* layers of VGG16. It would be interesting to incorporate this idea with RefineNet, while using the ZF Net [14] to study the impact on detector accuracy and run-time.

Results visualization: Fig. 3 demonstrates example images with the iterations of RefineNet visualized in different colors. RefineNet is shown to improve box localization on a variety of challenging cases, including small objects, partial truncation, and partial occlusion. We plot detection results at a low threshold for the analysis, leading to some false positives, but note that these are generally with a lower score than the true positives visualized. Fig. 4 demonstrates some challenging cases for RefineNet, mostly due to heavy occlusion. For instance, Fig. 4 depicts a case where RefineNet improves over the baseline but still does not fully localize an occluded vehicle (middle image). Another example is the many parked vehicles at close proximity, leading to a less localized box after the iterative refinement steps.

[1]Using Nvidia GTX Titan X

TABLE III

AUC ACHIEVED BY STATE-OF-THE-ART DETECTORS ON KITTI OBJECT DETECTION BENCHMARK AT MODERATE DIFFICULTY SETTINGS. ASTERISK (*) - METHOD EMPLOYS THE VGG16 NETWORK.

Detector	AUC			Runtime(sec)
	Moderate	Easy	Hard	
SubCNN* [16]	**89.04**	90.81	**79.27**	2
SDP* [19]	88.85	90.14	78.38	0.40
3DOP* [17]	88.64	**93.04**	79.10	3
Faster R-CNN* [8]	81.84	86.71	71.12	2
RefineNet (ours)	79.17	89.88	66.38	**0.22**
Regionlets [20]	76.45	84.75	59.70	1
SubCat [21]	75.46	84.14	59.71	0.7
3DVP [18]	75.77	87.46	65.38	40
OC-DPM [22]	65.95	74.94	53.86	10

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 upto **6%** in AUC. Our method relies on using already computed features making the detector very fast. Specifically, RefineNet runs in about **0.22** seconds per image. On KITTI object detection benchmark, it achieve 79.19% on moderate difficulty settings. It is the fastest detector that achieves upwards of 70% AUC. On easy difficulty settings, it achieves 90% AUC which is close to state-of-the-art result. The proposed approach was shown to greatly improve detection performance using the ZF architecture. Performance improvement using deeper networks, such as VGG, will be studied in the future.

VI. ACKNOWLEDGMENTS

The authors would like to thank the support of our associated industry partners, the reviewers for their constructive comments, and our colleagues at the Labotratory for Intelligent and Safe Automobiles for helpful discussions and assistance.

REFERENCES

[1] 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.

[2] T. Gandhi and M. M. Trivedi, "Image based estimation of pedestrian orientation for improving path prediction," in *IEEE Intelligent Vehicles Symposium*, 2008.

[3] 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.

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

[5] 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.

[6] 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.

[7] R. Girshick, "Fast r-cnn," in *International Conference on Computer Vision (ICCV)*, 2015.

Fig. 3. Sample images with 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. Note that the false positives shown in the last image have a low score of 90% and 85%. RefineNet is shown to improve localization in non-occluded and partially-occluded, truncated, and small object cases.

[8] 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 (NIPS)*, 2015.

[9] 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.

[10] 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.

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

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

[13] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *IEEE Conf. Computer Vision and Pattern Recognition*, 2012.

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

[15] 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 (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

Fig. 4. Sample images with detection boxes generated using RefineNet model M_2 on KITTI validation set. The images show challenging cases for RefineNet due to heavy occlusion.

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

[17] X. Chen, K. Kundu, Y. Zhu, A. Berneshawi, H. Ma, S. Fidler, and R. Urtasun, "3d object proposals for accurate object class detection," in *NIPS*, 2015.

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

[19] 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 *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2016.

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

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

[22] B. Pepik, M. Stark, P. Gehler, and B. Schiele, "Occlusion patterns for object class detection," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.