Supplementary Material for
Self-Supervised Object Detection via Generative Image Synthesis

Siva Karthik Mustikovela[1,3*] Shalini De Mello[1] Aayush Prakash[1]
Umar Iqbal[1] Sifei Liu[1] Thu Nguyen-Phuoc[2] Carsten Rother[3] Jan Kautz[1]
[1]NVIDIA [2]University of Bath [3]Heidelberg University

{siva.mustikovela, carsten.rother}@iwr.uni-heidelberg.de; aayush382.iitkgp@gmail.com;
T.Nguyen.Phuoc@bath.ac.uk; {shalinig, sifeil, uiqbal, jkautz}@nvidia.com

1. Overview

In this supplementary document, we provide the architectural and training details of our SSOD framework. Section 2 provides the architectures for the various networks ($\mathcal{S}$, $\mathcal{F}$, $\mathcal{D}_{scn}$, $\mathcal{D}_{fg}$, $\mathcal{D}_{bg}$) used in SSOD. Section 3 describes the training procedure and all its hyper-parameters. Further, in Section 4 we present the analysis of the case when the pose-aware synthesis network is trained directly on the target data with unknown number of objects per image, which fails to successfully disentangle the background and foreground representations. Lastly, we discuss the performance of the object detector on heavily occluded objects ('Hard' cases in KITTI [2]) in Section 5. We open-source our code at https://github.com/NVlabs/SSOD.

2. Network Architectures

2.1. Pose-Aware Synthesis Network

The architecture for the pose-aware synthesis network ($\mathcal{S}$), described in Section 3.3 and Figure 3 of the main paper, is detailed in Table 3. The architectures of $\mathcal{S}$'s foreground and background object branches are presented in Table 1 and Table 2. As shown in Figure 3 of the main paper, each of these branches take a learnable 3D object representation as input and perform a series of 3D convolutions on them. The 3D features are stylized using separate input style codes for the foreground (z_f) and the background (z_b) objects. The style codes pass through their respective MLP blocks, the output of which is used by adaptive instance normalization (AdaIN) [4] to control the style of each generated object/background. The MLP blocks contain 4 fully-connected layers each with a width of 200 neurons and leaky ReLU activations after each layer. The resulting 3D features are finally transformed using the input pose. These branches are used by $\mathcal{S}$ (Table 3), which performs an element-wise maximum operation on all the 3D features from all foreground objects and the background, followed by a projection of these features onto 2D. This is further followed by a set of 2D convolutions to generate an image of size 256×256.

*Siva Karthik Mustikovela was an intern at NVIDIA during the project.

2.2. Scene and MSO Discriminator

The architectures of scene ($\mathcal{D}_{scn}$) and multi-scale object ($\mathcal{D}_{mso}$) discriminators are described in Table 4. The input to each of these networks is an image of size 256×256. This is followed by a set of 2D convolutions and a fully-connected layer to produce a single class membership score indicating the probability of real/fake for the input image. We use Spectral Normalization [7] in these discriminator networks.

2.3. Patch Discriminator for Foreground and Background

The patch-based architectures of the foreground ($\mathcal{D}_{fg}$) and the background ($\mathcal{D}_{bg}$) discriminators are described in Table 5. The input to each of these networks is an image of size 256×256. Each network is fully convolutional and produces an output of size 8×8. We use Spectral Normalization [7] in these discriminator networks.

2.4. Object Detection Network

We use Faster-RCNN [9] with a Resnet-50-FPN [6] backbone as our detection network. It takes a 2D image as input and extracts features using the backbone layer. These features are further used by the object detection head to predict the top-left and bottom-right corners of the bounding box pertaining to a detected object. We use the object detection implementation from [11] in our work.

3. Training

We implement SSOD in Tensorflow [1]. We use the Adam [5] optimizer to learn our networks with beta parameter values (0.9, 0.99). The learning rate for all the networks is set to 0.00005. The batch size is 16. The dimensions of the style codes for the foreground (z_f) and the background (z_b) are 200 and 100, respectively. The style codes are sampled from a uniform distribution between (-1, 1). For our camera in $\mathcal{S}$ we use a focal length of $35mm$ with a sensor size of $32mm$ similar to [8].

Layer	Kernel Size	stride	Activation	Normalization	Output Dimension
Learnable 3D-Code	-	-	LReLU	AdaIN	$4 \times 4 \times 4 \times 512$
3D-Deconv	$3 \times 3 \times 3$	2	LreLU	AdaIN	$8 \times 8 \times 8 \times 128$
3D-Deconv	$3 \times 3 \times 3$	2	LreLU	AdaIN	$16 \times 16 \times 16 \times 64$
3D-Transform	-	-	-	-	$16 \times 16 \times 16 \times 64$

Table 1: **Architecture of foreground object branch**

Layer	Kernel Size	stride	Activation	Normalization	Output Dimension
Learnable 3D-Code	-	-	LReLU	AdaIN	$4 \times 4 \times 4 \times 256$
3D-Deconv	$3 \times 3 \times 3$	2	LreLU	AdaIN	$8 \times 8 \times 8 \times 128$
3D-Deconv	$3 \times 3 \times 3$	2	LreLU	AdaIN	$16 \times 16 \times 16 \times 64$
3D-Transform	-	-	-	-	$16 \times 16 \times 16 \times 64$

Table 2: **Architecture of background object branch**

3.1. Training Procedure

As described in the paper, we adopt a stage-wise training strategy to learn the modules of SSOD.

Uncoupled Training. Here, we first train the pose-aware synthesis network for 20 epochs. We initialize the weights of $\mathcal{S}$, $\mathcal{D}_{scn}$, $\mathcal{D}_{mso}$ using $\mathcal{N}(0, 0.2)$ and biases with 0. We train $\mathcal{S}$, supervised by the discriminators $\mathcal{D}_{scn}$ and $\mathcal{D}_{mso}$ in a Generative Adversarial Network [3] framework. We found empirically that updating $\mathcal{S}$ twice for every update of $\mathcal{D}_{scn}$ and $\mathcal{D}_{mso}$, results in the best visual quality. The weights for the losses from $\mathcal{D}_{scn}$ and $\mathcal{D}_{mso}$ are 0.5, each. The real images for $\mathcal{D}_{scn}$ and $\mathcal{D}_{mso}$ are sampled from the real-world source image collection $\{\mathbf{I}_s\}$. During this training stage, we synthesize images with a single foreground object as described in Sec. 3.3 of the main paper.

Next, we train the object detector $\mathcal{F}$ (initialized with ImageNet pretraining) using 10k images synthesized by $\mathcal{S}$ containing 1 or 2 objects paired with their computed bounding box labels $\langle \mathbf{I}_g, \mathbf{A}_g \rangle$. The learning rate for $\mathcal{F}$ is set to 0.00005 and it is trained for 10 epochs. In addition to the images synthesized from $\mathcal{S}$, we use real background regions $\{\mathbf{I}_t^b\}$ extracted from target collection $\{\mathbf{I}_t\}$. To obtain the real background images $\{\mathbf{I}_t^b\}$, we leverage Grad-CAM [10] to identify regions in $\{\mathbf{I}_t\}$ that do not contain the object of interest ('Vehicle', 'Car', 'Wagon', 'Van') with a high confidence (> 0.9). To achieve this, we use layer-4 of the Resnet-152 network trained on Imagenet with Grad-CAM.

Coupled Training. In this stage, we tightly couple all networks ($\mathcal{S}$, $\mathcal{F}$, $\mathcal{D}_{scn}$, $\mathcal{D}_{mso}$, $\mathcal{D}_{fg}$, $\mathcal{D}_{bg}$) together in an end-to-end manner and fine-tune them with the source $\{\mathbf{I}_s\}$, target $\{\mathbf{I}_t\}$ and synthesized $\{\mathbf{I}_g\}$ images. Similar to the uncoupled training stage, the real images for training $\mathcal{D}_{scn}$ and $\mathcal{D}_{mso}$ are sampled from the source collection $\{\mathbf{I}_s\}$. The real images for training $\mathcal{D}_{bg}$ come from $\{\mathbf{I}_t^b\}$.

The real images for training $\mathcal{D}_{fg}$ are obtained as follows. We use the object detector $\mathcal{F}$ trained in the uncoupled training stage to obtain high-confidence (> 0.9) detections of cars in $\{\mathbf{I}_t\}$. Further, we use these detections to crop out image patches $\{\mathbf{P}_t\}$ of size 256×256 around the object using the centers of the detections. These form image-annotation pairs $\langle P_t, M_t \rangle$ where M_t is the corresponding binary mask indicating the region inside the detection. M_t is used in computing the foreground appearance loss as discussed in Sec. 3.5.1 of the main paper.

The weights for the losses from $\mathcal{D}_{fg}$ and $\mathcal{D}_{bg}$, ($\mathcal{L}_{fg}$ and $\mathcal{L}_{bg}$) are initially set to a 0.05 and are progressively increased by 0.01 every 200 iterations to reach 0.5. $\mathcal{L}_{fg}$ and $\mathcal{L}_{bg}$ update the components of $\mathcal{S}$ that affect the overall appearance of images. Hence only the parameters of the MLPs and of 2D convolutional blocks of $\mathcal{S}$ are updated. $\mathcal{F}$ is trained using images synthesized by $\mathcal{S}$ using the image-annotation pairs $\langle I_g, A_g \rangle$. The weight for $\mathcal{L}_{det}$ is 0.4. We train SSOD in coupled training stage for 25k iterations.

4. Training synthesizer directly on target data

In this section, we explore the case when the pose-aware synthesis network, $\mathcal{S}$ is trained directly on the target data $\{\mathbf{I}_t\}$ instead of the source data $\{\mathbf{I}_s\}$. As described in Sec. 4.1 of the main paper, we use the Compcars dataset [12] as the source dataset $\{\mathbf{I}_s\}$, which is an in-the-wild collection of images with one car per image (see examples in Figure 1 of main paper). The target dataset for this experiment is the KITTI [2] dataset which contains outdoor driving scenes with unkown numbers of cars image (zero to multiple cars per image) with heavy occlusions, reflections and extreme

	Layer	Kernel Size	stride	Activation	Normalization	Output Dimension
3D Branches	$n\times$ FG Branch (a)	-	-	-	-	$16 \times 16 \times 16 \times 64$
	BG Branch (b)	-	-	-	-	$16 \times 16 \times 16 \times 64$
	Element-Wise Maximum(a,b)	-	-	-	-	$16 \times 16 \times 16 \times 64$
Project	Collapse	-	-	-	-	$16 \times 16 \times (16.64)$
	Conv	1×1	1	LReLU	-	$16 \times 16 \times 256$
2D Convs	2D-Deconv	4×4	2	LReLU	-	$32 \times 32 \times 128$
	2D-Deconv	4×4	2	LReLU	-	$64 \times 64 \times 64$
	2D-Deconv	4×4	2	LReLU	-	$128 \times 128 \times 32$
	2D-Deconv	4×4	2	LReLU	-	$256 \times 256 \times 16$
	2D-Conv (to RGB)	4×4	1	-	-	$256 \times 256 \times 3$

Table 3: **Architecture of pose-aware synthesis network**

Layer	Kernel Size	stride	Activation	Normalization	Output Dimension
2D-Conv (from RGB)	4×4	1	LReLU	-	$256 \times 256 \times 8$
2D-Conv	5×5	2	LReLU	SpectralNorm	$128 \times 128 \times 16$
2D-Conv	5×5	2	LReLU	SpectralNorm	$64 \times 64 \times 32$
2D-Conv	5×5	2	LReLU	SpectralNorm	$32 \times 32 \times 64$
2D-Conv	5×5	2	LReLU	SpectralNorm	$16 \times 16 \times 128$
2D-Conv	5×5	2	LReLU	SpectralNorm	$8 \times 8 \times 256$
2D-Conv	5×5	2	LReLU	SpectralNorm	$4 \times 4 \times 512$
Fully Connected	-	-	-	-	1

Table 4: **Architecture of scene discriminator and multi-scale object discriminators**

lighting (see examples in Figure 1).

To train $\mathcal{S}$ with the target data, we first identify regions in $\{\mathbf{I}_t\}$, which contain foreground objects of interest. We use Grad-CAM [10] to identify regions in the target image collections where there is a high confidence response for the classes ('Vehicle', 'Car', 'Wagon', 'Van'). We do this to obtain training images with a fixed number (one) car per image. Figure 1 illustrates some example images obtained using this method. However, notice that these images do not necessarily contain only one foreground object, but a random unknown number of them per image. This is because Grad-CAM can only identify regions, which have a high response to a specific class and cannot separate out objects. However, it is important for $\mathcal{S}$ to know the number of foreground objects per image as discussed in Sec. 3.3 of the main paper.

Since the number of foreground objects is not known *a priori* for these real-world images, it is not possible to correctly specify the number of foreground objects in the synthesizer while synthesizing images with it. This makes it difficult for $\mathcal{S}$ to disentangle foreground and background regions and learn separable representations for them during training with such data. Figure 2 presents images synthesized by $\mathcal{S}$ trained on foreground images extracted from the KITTI target image collection (Figure 1). Each row in this figure contains images with a fixed input foreground and background code. The horizontal translation value of the foreground objects is varying across the columns. Figure 2 shows that, even when the input translation of the foreground object varies, the image is constant throughout the columns without translation of the foreground object. This is because $\mathcal{S}$ is unable to disentangle the foreground and background regions and hence providing a different input

Layer	Kernel Size	stride	Activation	Normalization	Output Dimension
2D-Conv (from RGB)	4×4	1	LReLU	-	$256 \times 256 \times 8$
2D-Conv	4×4	2	LReLU	SpectralNorm	$128 \times 128 \times 16$
2D-Conv	4×4	2	LReLU	SpectralNorm	$64 \times 64 \times 32$
2D-Conv	4×4	2	LReLU	SpectralNorm	$32 \times 32 \times 64$
2D-Conv	4×4	2	LReLU	SpectralNorm	$16 \times 16 \times 128$
2D-Conv	4×4	2	LReLU	SpectralNorm	$8 \times 8 \times 256$
2D-Conv	1×1	1	LReLU	-	$8 \times 8 \times 256$

Table 5: **Architecture of foreground and background discriminators**

translation value to the GAN for the foreground object does not induce any changes in the generated images.

On the other hand, when $\mathcal{S}$ is trained using a source dataset $\{\mathbf{I}_s\}$ (where the number of objects per image is known) and adapted to a target dataset (KITTI [2]) using our approach, it is able to disentangle foreground and background representations. As a result, the pose of the foreground object can be controlled by the input parameters as shown in Figure. 3. This experiment shows that it is imperative to have access to an image collection $\{\mathbf{I}_s\}$ where the number of foreground objects per image is known a priori to be able to successfully train our controllable GAN network $\mathcal{S}$. In our case, we assume that we have access to an image collection with one object per image.

5. Limitations

In this section, we present a qualitative analysis of the objects detected by SSOD for images from the challenging KITTI [2] dataset as discussed in Sec. 4.5 of the main paper. These images contain objects with heavy occlusions, reflections and extreme lighting variations. In Figures 4, 5 we illustrate the 2D bounding boxes predicted by SSOD in green and the ground truth bounding boxes in blue. In all these images it can be seen that the cars are reliably detected under moderate occlusions, lighting variations and reflections. On the other hand, cars with extremely heavy occlusions where only a small part of the car is visible are sometimes missed. We observe that these are generally cases where cars are parked along the side of the road and are heavily occluded by each other. Such cases where there is heavy occlusion are classified as 'Hard' in the KITTI [2] dataset. We believe that this problem can be alleviated by specifically learning to model the layout, occlusions of objects and context of scenes in generated images to be similar to target data. For example, cars could occlude each other in parked scenarios or roads with heavy traffic. We consider this to be a future goal to address.

References

[1] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *OSDI*, pages 265–283, 2016. 1

[2] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 1, 2, 4, 5, 6, 7

[3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *NeurIPS*, volume 27. Curran Associates, Inc., 2014. 2

[4] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017. 1

[5] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 1

[6] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, July 2017. 1

[7] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *ICLR*, 2018. 1

[8] Thu Nguyen-Phuoc, Christian Richardt, Long Mai, Yong-Liang Yang, and Niloy Mitra. Blockgan: Learning 3d object-aware scene representations from unlabelled images. In *NeurIPS*, 2020. 1

[9] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015. 1

[10] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, Oct 2017. 2, 3, 5

[11] Yuxin Wu et al. Tensorpack. 2016. 1

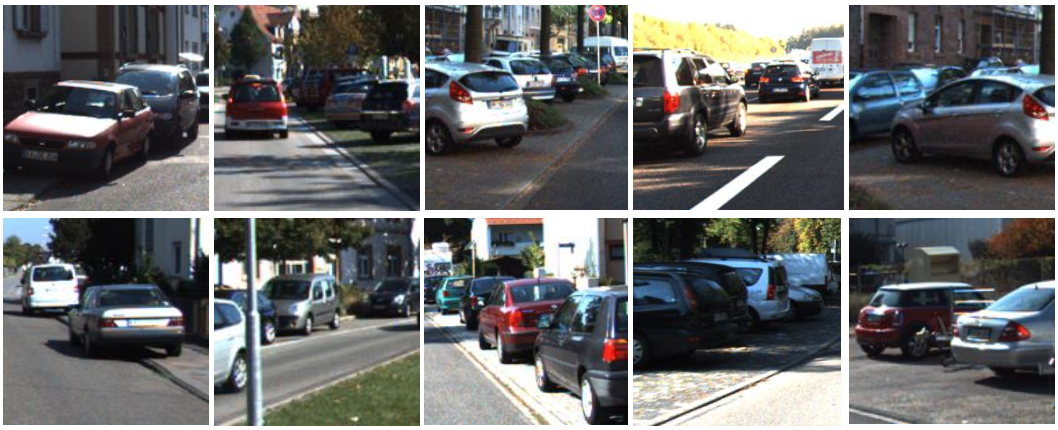

Figure 1: **Sample image from target data (KITTI).** The figure shows sample images from KITTI [2] data obtained for training the synthesis network $\mathcal{S}$. They are obtained by using Grad-CAM [10] on the original KITTI images and finding regions where there is a high confidence response for the classes 'Vehicle', 'Car', 'Wagon', 'Van'. It can be seen that each image contains a random number of multiple objects and not necessarily a single object.

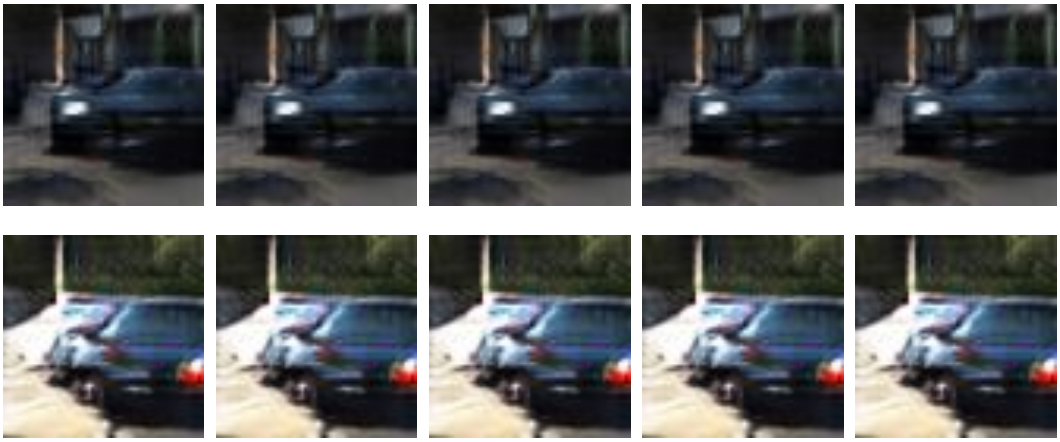

Figure 2: **Images synthesized from $\mathcal{S}$ trained on the target image collection from KITTI (Figure 1).** Each row in this figure contains images with a fixed input foreground and background code. The input horizontal translation of the foreground objects is varying across the columns. It can be seen that even when the input translation of the foreground object varies, the image is constant across the columns. This is because $\mathcal{S}$ is unable to disentangle the foreground and background regions and hence translating the foreground object does not induce any changes in the generated images.

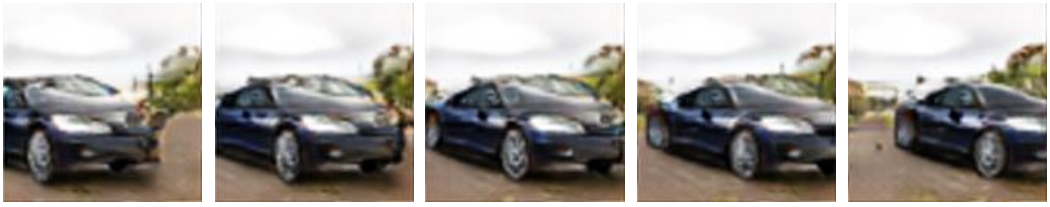

Figure 3: **Images synthesized from $\mathcal{S}$ trained on source image collection.** The input horizontal translation of the foreground object is varying from left to right. The object is translating along the horizontal axis according to the input translation, while the background remains constant. This is because on being trained with the source image collection $\mathcal{S}$ is able to disentangle the foreground and background representations.

[12] Linjie Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. A large-scale car dataset for fine-grained categorization and verification. In *CVPR*, 2015. 2

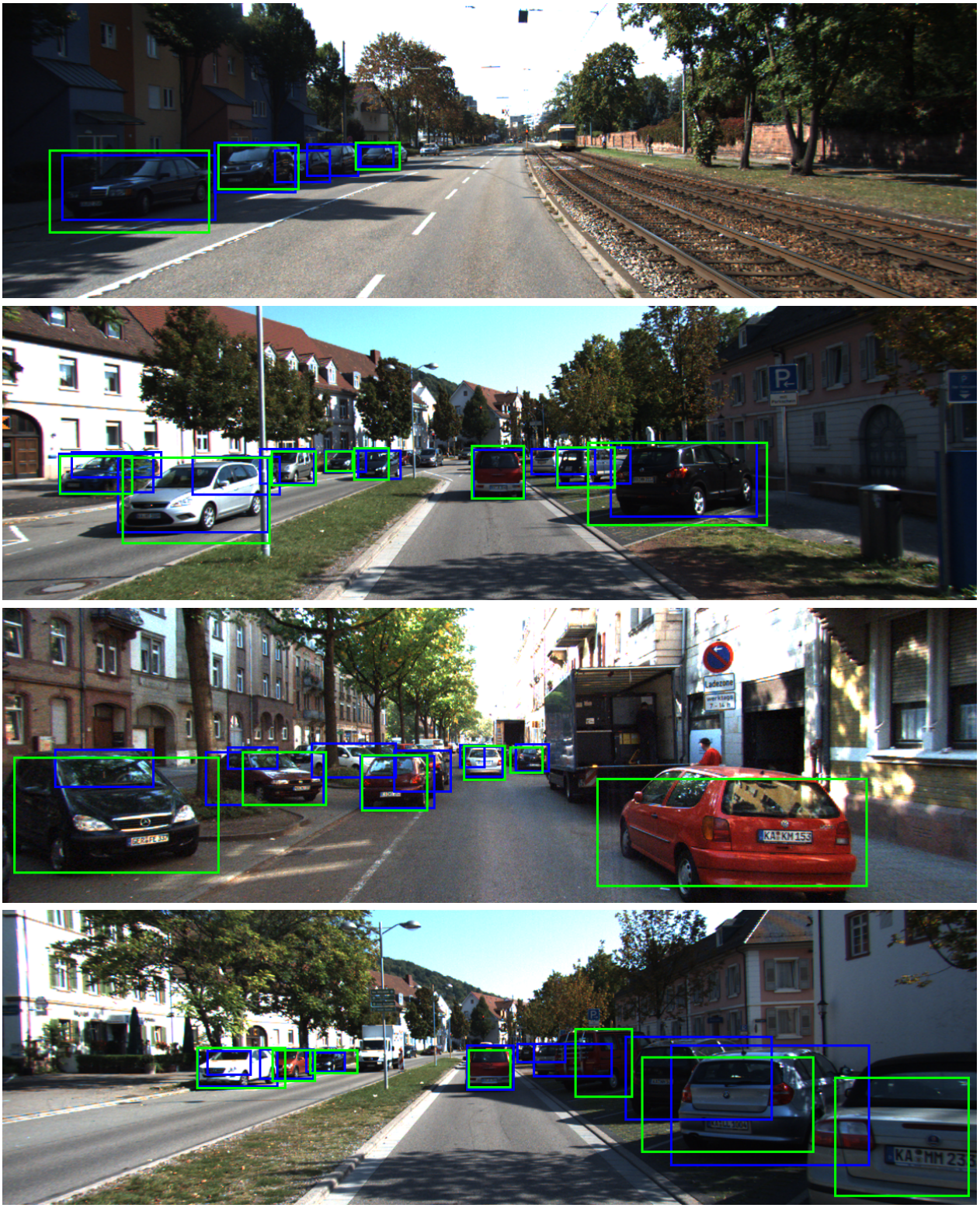

Figure 4: **Visualization of object detections by SSOD on the KITTI [2] dataset.** Green boxes show SSOD's predictions and Blue boxes show ground truth annotations. It can be seen that most of the cars with none to moderate occlusions are reliably detected. Cars with extremely heavy occlusions where only a small part of the car can be seen are sometimes missed.

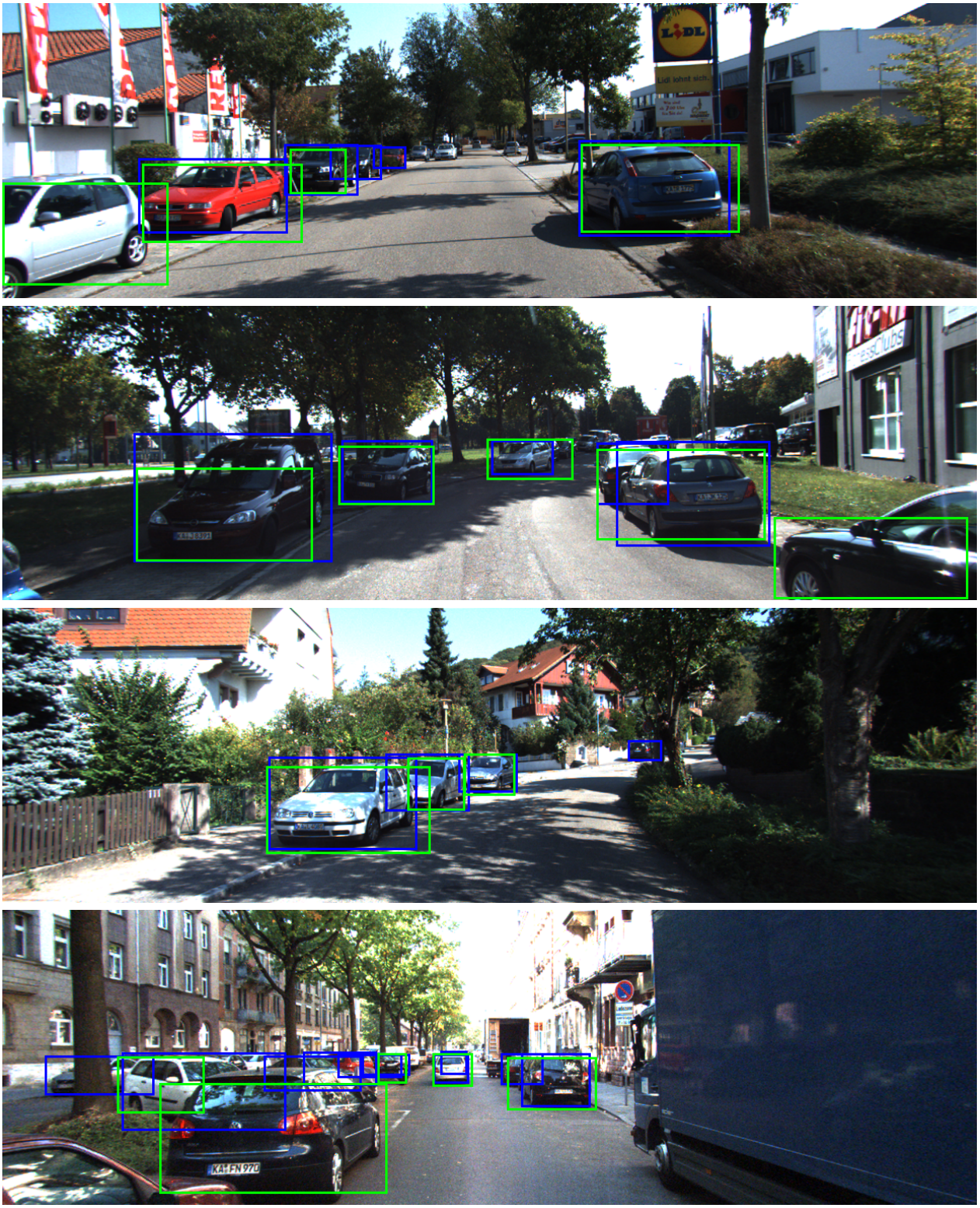

Figure 5: **Visualization of object detections by SSOD on the KITTI [2] dataset.** Green boxes show SSOD's predictions and blue boxes show ground truth annotations. It can be seen that most of the cars with none to moderate occlusions are reliably detected. Cars with extremely heavy occlusions where only a small part of the car can be seen are sometimes missed.