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

Abstract

We present SSOD – the first end-to-end analysis-by-synthesis framework with controllable GANs for the task of self-supervised object detection. We use collections of real-world images without bounding box annotations to learn to synthesize and detect objects. We leverage controllable GANs to synthesize images with pre-defined object properties and use them to train object detectors. We propose a tight end-to-end coupling of the synthesis and detection networks to optimally train our system. Finally, we also propose a method to optimally adapt SSOD to an intended target data without requiring labels for it. For the task of car detection, on the challenging KITTI and Cityscapes datasets, we show that SSOD outperforms the prior state-of-the-art purely image-based self-supervised object detection method Wetectron. Even without requiring any 3D CAD assets, it also surpasses the state-of-the-art rendering-based method Meta-Sim2. Our work advances the field of self-supervised object detection by introducing a successful new paradigm of using controllable GAN-based image synthesis for it and by significantly improving the baseline accuracy of the task. We open-source our code at https://github.com/NVlabs/SSOD.

1. Introduction

Object detection plays a crucial role in various autonomous vision pipelines, *e.g.*, in robotics and self-driving. Convolutional neural networks-based detection methods, such as [40, 32], have achieved impressive performance. However, they are fully-supervised and require large amounts of human annotated data, which is time-consuming to acquire for all object types and operating environments. They also do not scale well when target domains change, *e.g.*, from one city to another in self-driving.

To reduce annotations, some existing works train detectors without requiring bounding box annotations and follow two paradigms. The first is of self/weakly supervised object detection methods [41, 42, 53], which either use image-level object presence labels (a.k.a. self-supervision) or point/scribble annotations (a.k.a weak-supervision). They also rely on high-quality object proposals detected by methods requiring human annotations [57]. The second paradigm is of rendering-based methods, including Meta-Sim [26] and Meta-Sim2 [10], which learn object detection from synthetically rendered images. Creating them, however, requires large collections of high-quality 3D CAD models for all the objects in the scene, manual scene setups and expensive rendering engines. Such images also tends to have a large domain gap from real-world ones.

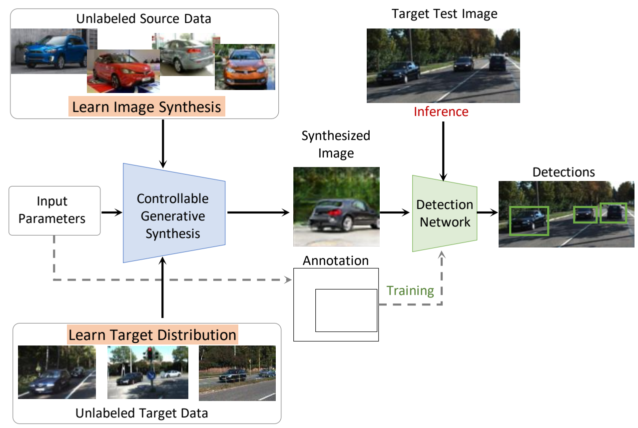

Figure 1. **Self-Supervised Object Detection.** We learn object detection purely using natural image collections without bounding box labels. We leverage controllable GANs to synthesize images and to detect objects together in a tightly coupled framework. We learn image synthesis from unlabeled singe-object source images (*e.g.*, Compcars [52]) and optimally adapt our framework to any multi-object unlabeled target dataset (*e.g.*, KITTI [15]).

Recently, there has been much progress in making Generative Adversarial Networks (GANs) [16] controllable using input parameters like shape, viewpoint, position and keypoints [36, 37, 38, 45], opening up the possibility of synthesizing images with desired attributes. Controllable GANs have also been used successfully to learn other vision tasks, *e.g.*, viewpoint [34] and keypoints [50, 56, 22] estimation in a self-supervised manner, but have not been explored previously for self-supervised object detection.

Inspired by these, we propose the first end-to-end analysis-by-synthesis framework for self-supervised object detection using controllable GANs, called SSOD (Fig. 1). We learn to both synthesize images and detect objects purely using unlabeled image collections, *i.e.*, without requiring bounding box-labels and without using 3D CAD

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

assets – a multi-faceted challenge not addressed previously. We learn a generator for object image synthesis using real-world single-object image collections without bounding box labels. By leveraging controllable GANs, which provide control over the 3D location and orientation of an object, we also obtain its corresponding bounding box annotation. To optimally train SSOD, we tightly couple the synthesis and detection networks in an end-to-end fashion and train them jointly. Finally, we learn to optimally adapt SSOD to a multi-object target dataset, also without requiring labels for it and improve accuracy further.

We validate SSOD on the challenging KITTI [15] and Cityscapes [8] datasets for car object detection. SSOD outperforms the best prior image-based self-supervised object detection method Wetectron [42] with significantly better detection accuracy. Furthermore, even without using any 3D CAD assets or scene layout priors it also surpasses the best rendering-based method Meta-Sim2 [10]. To the best of our knowledge, SSOD is the first work to explore using controllable GANs for fully self-supervised object detection. Hence, it opens up a new paradigm for advancing further research in this area. SSOD significantly outperforms all competing image-based methods and serves as a strong baseline for future work.

To summarize, our main contributions are:

- We propose a novel self-supervised object detection framework via controllable generative synthesis, which uses only image collections without any kind of bounding box annotations.
- We propose an end-to-end analysis-by-synthesis framework, which can optimally adapt the synthesizer to both the downstream task of object detection and to a target dataset in a purely self-supervised manner.
- Our experiments on two real-world datasets show ∼2x performance improvement over SOTA image-based self-supervised object detection methods. Also, without using 3D CAD assets, SSOD outperforms on average, the rendering-based baseline of Meta-Sim2 [10].

2. Related Work

Self-supervised task learning. Several recent works attempt to learn a variety of 2D and 3D computer vision tasks in a self-supervised manner. In 2D computer vision, several works tackle the problem of object keypoint estimation [50, 56, 22] and part segmentation [20, 7]. [3] obtains an object mask along with the generated image. However, there is no control over the pose and style of the generated object. Alongside, in 3D computer vision, there are several attempts to learn object reconstruction [24, 29, 31, 30], viewpoint estimation [34] and point cloud estimation [35]. These works present interesting approaches to address their respective problems for single object images, but do not address multi-object analysis.

Concurrently, there has also been tremendous progress in high-quality controllable generative synthesis using learned 3D object representations [37, 36, 38, 45, 11] or implicit representations [33, 54, 44]. Some of these works have been used in analysis-by-synthesis frameworks to solve computer vision tasks, including 3D reconstruction [31, 30, 18, 17], viewpoint estimation [34] and keypoint estimation [22]. However, no prior work explores self-supervised object detection via controllable GANs and we are the first work to do so.

Weakly supervised object detection. Recent works also address the problem of self-supervised object detection using only a collection of images and image-level tags of object presence. Such methods pose the problem either in a multiple instance [4, 49, 53, 41, 14], discriminative [47], curriculum [55, 25, 42] or self-taught [23] learning framework. However, such methods rely heavily on object proposals generated by methods like [57, 1, 51], which, themselves, need low-level edge-based annotations from humans. Additionally, they also cannot modify or control input images according to the requirements of the detector or a target dataset. In contrast we learn a controllable synthesis module, to synthesize images that maximize the detector's performance on a target dataset.

Learning object detection from synthetic data. Works like [43, 6, 13, 26, 10, 39] learn object detection through synthetic data from graphics renderers. [43] obtains synthetic images and annotations from a game engine. In [6, 13], the authors exactly mimic real world dataset (*e.g.*, KITTI [15]) in a synthetic simulator. In [39], the authors synthesize scenes by randomizing location, orientations and textures of objects of interest in a scene. In Meta-Sim [26] and Meta-Sim2 [10], the authors propose a strategy to learn optimal scene parameters to generate images similar to a target dataset. While methods like [6, 13] use annotations from real world datasets to mimic the datasets in synthetic worlds, other methods like [43, 10, 39] generate synthetic data without using any real world annotations. While these approaches learn only from rendered data, they require 3D CAD models of objects and scenes along with rendering setups, both of which are expensive to acquire. Moreover, graphics renderers are often not differentiable making it difficult to learn and propagate gradients through them for learning a downstream task. Also, synthetic data introduces a domain gap with respect to real target data both in terms of appearance and layout of scenes that affects detection accuracy. In contrast, our goal is to learn both data generation and object detection from real-world images without bounding box annotations and without requiring 3D CAD models or rendering setups. Our GAN-based framework allows us to adapt to the distribution of the target data and synthesize data that is optimal for the downstream task.

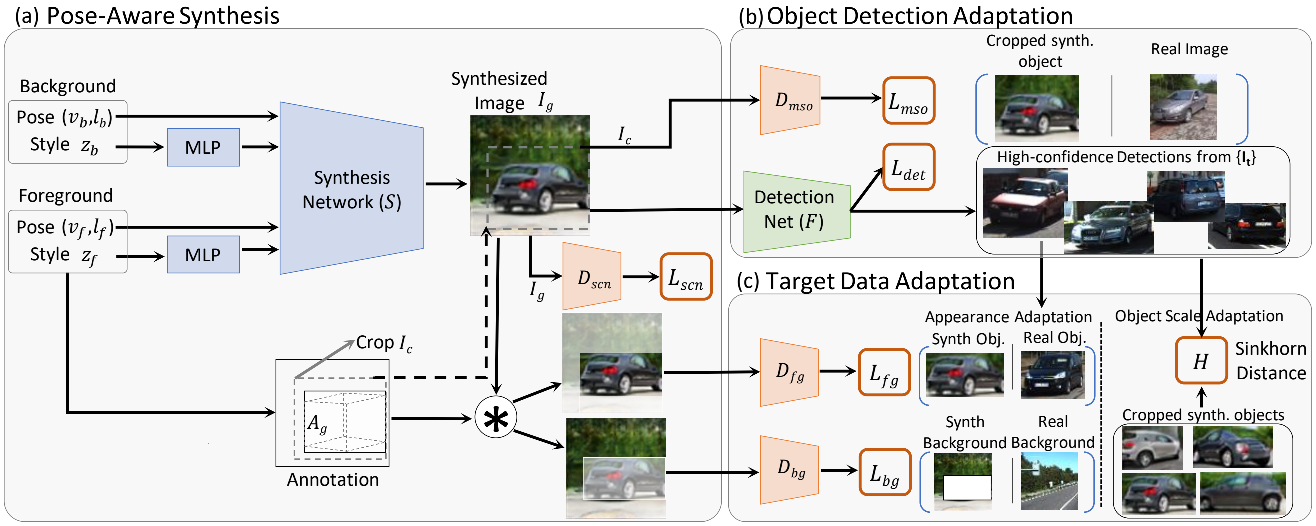

Figure 2. **Overview of Self-Supervised Object Detection.** SSOD contains three modules: (a) a pose-aware synthesis module that generates images with objects in pre-defined poses using a controllable GAN for training object detectors; (b) an object detection adaptation module that guides the synthesis process to be optimal for the downstream task of object detection and the (c) a target data adaption module that helps SSOD to adapt optimally to a target data distribution. We train all modules in a tightly-coupled end-to-end manner.

3. Self-Supervised Object Detection

3.1. Problem Setup

Our goal is to learn a detection network $\mathcal{F}$, which best detects objects (*e.g.*, cars) in a target domain (*e.g.*, outdoor driving scenes from a city). We further assume that we have available to us an unlabeled image collection $\{\mathbf{I}_t\}$ from the target domain each containing an unknown number of objects per image (see examples in Fig. 1). To train $\mathcal{F}$, we leverage object images and their bounding box annotations synthesized by a controllable generative network $\mathcal{S}$, which, in turn, is also learnt using unlabeled object collections. Specifically, to learn $\mathcal{S}$, we use an additional sufficiently large unlabeled (bounding box annotation free) single-object source collection $\{\mathbf{I}_s\}$, containing images with only one object per image, but not necessarily from the target domain where the detector must operate (see examples in Fig. 1). We discuss more about the need for this assumption in Sec 3.3. We train our system with both $\{\mathbf{I}_t\}$ and $\{\mathbf{I}_s\}$, and evaluate it on a held-out labeled *validation* set from the target domain, which is disjoint from $\{\mathbf{I}_t\}$ and is never used for training.

3.2. Overview of SSOD

We present an overview of SSOD in Fig. 2. It contains three modules: (a) a pose-aware synthesis; (b) an object detection adaptation and (c) a target data adaption module.

The pose-aware synthesis module (Fig. 2(a)) contains a controllable synthesis network $\mathcal{S}$. We model $\mathcal{S}$ by a pose-aware generator, which synthesizes images $\{\mathbf{I}_g\}$ of objects conditioned on the pose parameters (viewpoint ($\boldsymbol{v}$) and location ($\boldsymbol{l}$)) and obtain 2D bounding box annotations $\{\mathbf{A}_g\}$ for them. Using the synthesized image-annotation pairs $\langle \mathbf{I}_g, \mathbf{A}_g \rangle$, along with images from $\{\mathbf{I}_t\}$, we train the object detector $\mathcal{F}$. The object detection adaptation module (Fig. 2(b)) is designed to provide feedback to the synthesis network $\mathcal{S}$ to optimally adapt it to the downstream task of object detection. It tightly couples the object detector $\mathcal{F}$ and synthesizer $\mathcal{S}$ for joint end-to-end training and also introduces specific losses to guide the synthesis process towards better object detection learning.

Lastly, the target data adaptation module (Fig. 2(c)) helps reduce the domain gap between the images synthesized by $\mathcal{S}$ and those in the target domain $\{\mathbf{I}_t\}$. It does so by introducing a set of spatially localized discriminative networks, which adapt the synthesis network $\mathcal{S}$ towards generating images closer to the target data distribution in terms of overall image appearance and scale of objects.

We train SSOD in two stages – uncoupled and coupled. During uncoupled training, we pre-train the synthesis network $\mathcal{S}$ on $\{\mathbf{I}_s\}$ without feedback from other modules. Next, we synthesize image-annotation pairs with $\mathcal{S}$ and use them along with $\{\mathbf{I}_t\}$ to pre-train $\mathcal{F}$. During the next coupled training phase, we jointly fine-tune SSOD's modules with both the source $\{\mathbf{I}_s\}$ and target $\{\mathbf{I}_t\}$ images, and the data synthesized by $\mathcal{S}$. We alternatively train $\mathcal{S}$ in one iteration and all other networks in the next one. We describe all the modules of SSOD in detail in the following sections.

3.3. Pose-Aware Synthesis

Our pose-aware synthesis network $\mathcal{S}$ is inspired by the recent BlockGAN [37], which has several desirable properties for object detection. It allows control over style, pose and number of objects in the scene by disentangling the background and foreground objects. Its architecture is illustrated in Fig. 3. To make BlockGAN [37] amenable to target data adaptation, we augment it with MLP blocks which learn to modify style vectors for both the foreground and background before they are input to the generator, such that the synthesized images are closer to the target dataset (Fig. 3).

The synthesis network $\mathcal{S}$ generates a scene I_g containing the foreground object in the specified location and orientation. The network contains category specific learnable canonical 3D codes for foreground and background objects, which are randomly initialized and updated during training. The 3D latent code of each object is passed through a corresponding set of 3D convolutions where the style of the object is controlled by input 1D style code vectors (from a uniform distribution) $\boldsymbol{z}_f$ for the foreground and $\boldsymbol{z}_b$ for the background through AdaIN (Fig. 3). These 3D features are further transformed using their input poses $(\boldsymbol{v}_f, \boldsymbol{l}_f)$ for one or more foreground objects. The value of $\boldsymbol{v}_f$ represents azimuth of the object and $\boldsymbol{l}_f$ represents its horizontal and depth translation. Each object is processed separately in its own 3D convolution branch. The resulting 3D features of all objects are collated using an element-wise maximum operation and then projected onto 2D using a perspective camera transformation followed by a set of 2D convolutions to yield I_g. The original BlockGAN [37] generates images at a resolution of 64×64. For our $\mathcal{S}$, we modify it and adopt the strategy of progressive growing of GANs [28, 27] to increase its synthesis resolution to 256×256.

We train $\mathcal{S}$ in a GAN setup with an adversarial loss [2] $\mathcal{L}_{scn}$ computed using a scene discriminator $\mathcal{D}_{scn}$ as:

$$\mathcal{L}_{scn} = -\mathbb{E}_{I_g \sim p_{\text{synth}}}[\mathcal{D}_{scn}(I_g)], \tag{1}$$

where $\mathcal{D}_{scn}(I_g)$ is the class membership score predicted by the scene discriminator $\mathcal{D}_{scn}$ for a synthesized image. This is one among several losses that we use to train $\mathcal{S}$. The real images input to $\mathcal{D}_{scn}$ are sampled from $\{\mathbf{I}_s\}$.

To train $\mathcal{S}$, we use a large set of real images with fixed and known (n) number of objects in each real image $\{\mathbf{I}_s\}$ without any requirement of bounding box annotations. Since we know n (in our case one object per image), while training $\mathcal{S}$ we can synthesize images with the same number of objects to pass to the discriminator, making it easier to train the generator. Having a single object image collection is not a requirement to train $\mathcal{S}$ and it has been shown in [37] that it can be trained successfully with 2 or more objects per image. However, having a large image collection $\{\mathbf{I}_s\}$ with known number of objects is crucial for training $\mathcal{S}$. Our attempts to train it with a target image collection $\{\mathbf{I}_t\}$ of city driving scenes, e.g. KITTI, with unknown number of objects per image were unsuccessful (details in supplementary material Sec. 4).

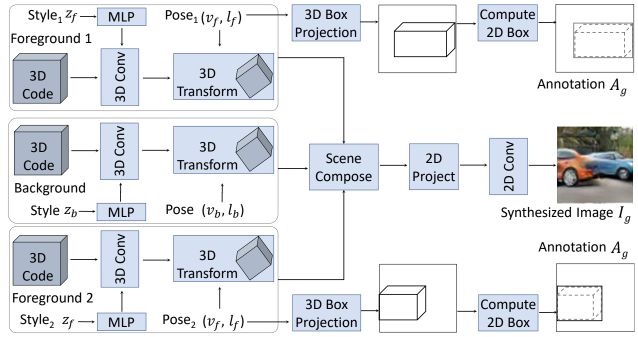

Figure 3. **Pose-Aware Synthesis Network ($\mathcal{S}$) Overview.** $\mathcal{S}$ takes as input separate style codes ($\boldsymbol{z}$) and poses ($\boldsymbol{v}$, $\boldsymbol{l}$) for the background and one/more foreground objects; transforms their respective learned 3D codes with the provided poses; and synthesizes images after passing them through several 3D convolutional, 2D projection and 2D convolutional layers. We use the provided poses to compute 2D bounding box labels for the synthesized objects.

3.3.1 Obtaining Bounding Box Annotations

The synthesis network $\mathcal{S}$ can generate a foreground object using a pose $(\boldsymbol{v}_f, \boldsymbol{l}_f)$. This key property allows us to localize the object in the synthesized image and to create a 2D bounding box (BBox) annotation for it. We use the mean 3D bounding box (in real-world dimensions) of the object class and project it forward onto the 2D image plane using $\mathcal{S}$'s known camera matrix and the object's pre-defined pose $(\boldsymbol{v}_f, \boldsymbol{l}_f)$ via perspective projection. The camera matrix is fixed for all synthesized images. We obtain the 2D bounding box A_g for the synthesized image I_g by computing the maximum and minimum coordinates of the projected 3D bounding box in the image plane. This procedure is illustrated in Fig. 3. The paired data $\langle I_g, A_g\rangle$ can then be used to train the object detection network $\mathcal{F}$.

3.4. Object Detection Adaptation

We introduce a set of objectives, which supervise $\mathcal{S}$ to synthesize images that are optimal for learning object detectors. These include an (a) object detection loss and (b) a multi-scale object synthesis loss, which we describe next.

3.4.1 Object Detection Loss

In our setup, we tightly couple the object detection network $\mathcal{F}$ to $\mathcal{S}$ such that it provides feedback to $\mathcal{S}$ (Fig. 2(b)). The object detection network $\mathcal{F}$ is a standard Feature Pyramid Network [32], which takes 2D images as input and predicts bounding boxes for the object. It is trained using the standard object detection losses ($\mathcal{L}_{det}$) [32]. While training SSOD, we compute the object detection loss $\mathcal{L}_{det}$ for the image-annotation pairs $\langle \mathbf{I}_g, \mathbf{A}_g\rangle$ synthesized by $\mathcal{S}$ and use it as an additional loss term for updating the weights of $\mathcal{S}$.

3.4.2 Multi-scale Object Synthesis Loss

It is important for $\mathcal{S}$ to be able to synthesize high quality images at varied object depths/scales, such that $\mathcal{F}$ can be optimally trained with diverse data. Hence, to extend the range of depths for which $\mathcal{S}$ generates high-quality objects, we introduce a multi-scale object synthesis loss, $\mathcal{L}_{mso}$ (Fig. 2(b)). To compute it, we use a synthesized image I_g's bounding box A_g and crop (in a differentiable manner) an image I_c using a dilated version of A_g with a unit aspect ratio such that the context around the object is considered. Further, we resize I_c to 256×256. We then pass I_c to a multi-scale object discriminator $\mathcal{D}_{mso}$. This makes the generated images match the appearance of the real images, with less surrounding background and simultaneously improves image quality. The real images input to $\mathcal{D}_{mso}$ are images from the source collection $\{\mathbf{I}_s\}$, also of size 256×256. The multi-scale object synthesis loss, $\mathcal{L}_{mso}$ is then given by:

$$\mathcal{L}_{mso} = -\mathbb{E}_{I_c \sim p_{\mathrm{synth}}}[\mathcal{D}_{mso}(I_c)], \tag{2}$$

where $\mathcal{D}_{mso}(I_c)$ is the realism score predicted by $\mathcal{D}_{mso}$ for the image crop I_c.

3.5. Target Data Adaptation

We train $\mathcal{S}$ with single-object images $\{\mathbf{I}_s\}$ acquired from a collection, which do not necessarily come from the final target domain. Hence, there may be a domain gap between the images synthesized by $\mathcal{S}$ and those from the target domain (see examples in Fig. 1 and Fig. 4). This makes $\mathcal{F}$, trained on images synthesized by $\mathcal{S}$, perform sub-optimally on the target domain. To address this, we introduce a target data adaptation module (Fig. 2(c)), whose focus is to adapt $\mathcal{S}$ such that it can synthesize images closer to the target data distribution. It uses foreground and background appearance losses to supervise training of $\mathcal{S}$, which make the synthesized images match the target domain. Additionally, it contains an object scale adaption block to match the scale of synthesised objects to the ones in the target domain. We align the synthesized data to the distribution of the target dataset without using any bounding box annotations. We describe these various components in detail.

3.5.1 Foreground Appearance Loss

We compute the foreground appearance loss $\mathcal{L}_{fg}$ via a patch-based [21] discriminator $\mathcal{D}_{fg}$ (Fig. 2(c)). It takes the synthesized image-annotation pair $\langle I_g, A_g \rangle$ as input and predicts a 2D class probability map, $\hat{c}_{fg} = \mathcal{D}_{fg}(I_g)$, where $\hat{c}_{fg}$ is the patch-wise realism score for the synthesized image I_g. The foreground appearance loss ($\mathcal{L}_{fg}$) for the synthesis network $\mathcal{S}$ is given by:

$$\mathcal{L}_{fg} = -\mathbb{E}_{I_g \sim p_{\mathrm{synth}}}[\hat{c}_{fg}] * M_g, \tag{3}$$

where $*$ indicates element-wise multiplication. M_g masks the loss to be computed only for the foreground region of the synthesized image. The real images used to train this discriminator come from the target collection $\{\mathbf{I}_t\}$. We acquire them by using the pre-trained object detection network $\mathcal{F}$ created during the first phase of uncoupled training (described in Sec. 3.2). Specifically, we infer bounding boxes for the images in the target dataset $\{\mathbf{I}_t\}$ using the pre-trained $\mathcal{F}$ and select a subset of images $\{\mathbf{P}_t\}$ with detection confidence >0.9. This forms an image-annotation pair $\langle P_t, M_t \rangle$, where M_t is the corresponding binary mask for the detected foreground objects in image P_t. The loss for training the discriminator $\mathcal{D}_{fg}$ is computed as:

$$\mathcal{L}_{d_{fg}} = -\mathbb{E}_{I_t \sim p_{\mathrm{real}}}[c_t] * M_t + \mathbb{E}_{I_g \sim p_{\mathrm{synth}}}[\hat{c}_{fg}] * M_g, \tag{4}$$

where c_t is the patch-wise classification score predicted by $\mathcal{D}_{fg}$ for a real image.

3.5.2 Background Appearance Loss

The background discriminator $\mathcal{D}_{bg}$ is also a patch-based discriminator (Fig. 2(c)), which predicts the realism of the background region in I_g with respect to the target data $\{\mathbf{I}_t\}$. We compute the background mask by inverting the binary foreground mask M_g. The background appearance loss for the synthesis network, $\mathcal{S}$ is given by

$$\mathcal{L}_{bg} = -\mathbb{E}_{I_g \sim p_{\mathrm{synth}}}[\hat{c}_{bg}] * (1 - M_g), \tag{5}$$

where $\hat{c}_{bg} = \mathcal{D}_{bg}(I_g)$ predicts the patch-wise realism score for the background region of the generated image.

The real images used to train $\mathcal{D}_{bg}$ are obtained by identifying patches in the target collection $\{\mathbf{I}_t\}$ where no foreground objects are present. To this end, we leverage pre-trained image classification networks and class-specific gradient-based localization maps using Grad-CAM [46]. Through this, we identify patches $\{\mathbf{I}_t^b\}$ in the target collection $\{\mathbf{I}_t\}$ that do not contain the object of interest. They serve as real samples of background images used to train $\mathcal{D}_{bg}$. The loss for training $\mathcal{D}_{bg}$ is computed as:

$$\mathcal{L}_{d_{bg}} = -\mathbb{E}_{I_t^b \sim p_{\mathrm{real}}}[c_t^b] + \mathbb{E}_{I_g \sim p_{\mathrm{synth}}}[\hat{c}_{bg}] * (1 - M_g), \tag{6}$$

where c_t^b is the patch-wise classification score predicted by $\mathcal{D}_{bg}$ for a real image.

With $\mathcal{L}_{fg}$ and $\mathcal{L}_{bg}$ we only update the components of $\mathcal{S}$ that affect its style and appearance. These include (a) the parameters of the MLP blocks (Fig. 3), which modify the foreground and background style codes and (b) the weights of 2D convolution layers. The foreground and background patches are obtained from the synthesized images using the annotations computed by our method (Sec. 3.3.1). Empirically, we observe this is effective enough in learning the foreground and background distributions of the target domain.

3.5.3 Object Scale Adaptation

We also find the optimal set of the object depth parameters that should be input into $\mathcal{S}$ to achieve the best performance on the target domain via this module. To this end, we use $\mathcal{S}$ to synthesize image-annotation pairs $\langle I_g^{d_r}, A_g^{d_r} \rangle$ for multiple different object depth ranges Θ=$\{d_r\}$ and also obtain $\{\alpha^{d_r}\}$, which is the collection of cropped synthesized objects. Depth d is one of the components of the location parameter l used to specify the synthesized object's pose. We sample depth values uniformly within each depth range d_r. For each depth range d_r, we train a detector $\mathcal{F}^{d_r}$ with its corresponding synthetic data $\langle I_g^{d_r}, A_g^{d_r}\rangle$. We use $\mathcal{F}^{d_r}$ to detect all object bounding boxes $\{\beta^{d_r}\}$ in the target collection $\{\mathbf{I}_t\}$, which have confidence >0.85. Finally we compute the optimal input depth interval for synthesis as:

$$d_o = \underset{d_i}{\operatorname{argmin}}\, \mathcal{H}(\Phi(\alpha^{d_i}), \Phi(\beta^{d_i})), \tag{7}$$

where Φ computes the conv5 features of a pre-trained image classification VGG [48]) network and $\mathcal{H}$ is the Sinkhorn distance [9] between the two feature distributions. We use a single corresponding detector trained with the optimum depth range d_o for the final evaluation on the target test data.

3.6. Training Procedure

We adopt a stage-wise training strategy to learn SSOD.

Uncoupled Training. We first pre-train $\mathcal{S}$ and $\mathcal{F}$ separately. We train the generator $\mathcal{S}$, supervised by the discriminators D_{scn} and D_{mso}, using the source collection $\{\mathbf{I}_s\}$ only. We then synthesize images with $\mathcal{S}$ containing 1 or 2 objects and compute their labels. We use them, along with real background regions extracted from the target data $\{\mathbf{I}_t^b\}$ using Grad-CAM [46] (described in Sec. 3.5.2) to pre-train $\mathcal{F}$.

Coupled Training. During this stage we couple all the networks together in an end-to-end manner and fine-tune them together with source $\{\mathbf{I}_s\}$ and target $\{\mathbf{I}_t\}$ collections, and the data synthesized by $\mathcal{S}$. We also adapt SSOD to the target data in this stage. We use a GAN-like training strategy and alternatively train $\mathcal{S}$ in one iteration and all other networks $\mathcal{D}_{scn}$, $\mathcal{F}$, $\mathcal{D}_{mso}$, $\mathcal{D}_{fg}$ and $\mathcal{D}_{bg}$ in the next one. Here $\mathcal{S}$ is supervised by all other modules and the total loss for training it is:

$$\begin{aligned}\mathcal{L}_{syn} =& \lambda_{scn}\,\mathcal{L}_{scn} + \lambda_{mso}\,\mathcal{L}_{mso} + \lambda_{det}\,\mathcal{L}_{det} \\ &+ \lambda_{fg}\,\mathcal{L}_{fg} + \lambda_{bg}\,\mathcal{L}_{bg},\end{aligned} \tag{8}$$

where $\{\lambda_i\}$ are the relative weights of the various losses. Lastly, as discussed in Sec. 3.5.3 we find the optimal set of input object depth parameters for $\mathcal{S}$ that align synthesized data further to the target distribution.

4. Experiments

We validate SSOD for detecting "car" objects in outdoor driving scenes. We assess quantitative performance using the standard mean Average Precision (mAP) metric at an Intersection-Over-Union (IOU) of 0.5. We provide network architecture and training details in the supplementary.

4.1. Datasets and Evaluation

We use three datasets containing images of car objects to train and evaluate SSOD: (a) the Compcars dataset [52] as the single-car source dataset and (b) two multi-car KITTI [15] and Cityscapes [8] target datasets containing outdoor driving scenes. During training, we do not use bounding box annotations for any of these datasets.

Compcars. The Compcars dataset [52] is an in-the-wild collection of 137,000 images with one car per image. It provides good diversity in car appearances, orientations and moderate diversity in scales (see examples Fig. 1). We use it as the source image collection $\{\mathbf{I}_s\}$ to train our controllable viewpoint-aware synthesis network $\mathcal{S}$.

KITTI. The challenging KITTI [15] dataset contains 375×1242 sized outdoor driving scenes with zero or multiple cars per image with heavy occlusions, reflections and extreme lighting (see examples in Fig. 1). We use it as one of our target datasets $\{\mathbf{I}_t\}$. We split it into disjoint *training* (6000 unlabeled images) and *validation* (1000 labeled images) sets. We report the mAP for Easy, Medium and Hard and all cases [15] of the its *validation* set.

Cityscapes. Similarly to KITTI, we also evaluate SSOD on the challenging Cityscapes [8] outdoor driving target dataset with images of size 512×1024. We use the version provided by [12] containing bounding box annotations. We split it into disjoint *training* (3000 unlabeled images) and *validation* (1000 labeled images) sets as provided in [12].

4.2. Ablation Study

We conduct ablation studies on the KITTI dataset to evaluate the contribution of each individual component of SSOD (Table 1). We evaluate object detection performance using mAP, and compute SinkHorn [9], KID [5] and FID [19] scores to compare the appearance of the synthesized foreground objects to objects in KITTI.

Quality of annotations Firstly, we estimate the accuracy of annotations obtained from our pipeline. For 260 images synthesized by the generator, we manually annotate the bounding boxes and measure the mAP between them and the annotations by our pipeline. It is 0.95 at an IoU of 0.5, which is reasonable for learning object detectors.

Uncoupled Training. We evaluate the efficacy of simply training the object detector $\mathcal{F}$ with images synthesized by $\mathcal{S}$, when each of these networks is trained separately without coupling. We compare the original BlockGAN [37] with an image resolution of 64×64 to two of its variants with image resolutions 128×128 and 256×256 that we train as described in Sec. 3.3. The results are shown in the top three rows of Table 1. They indicate that synthesized foreground objects at higher resolutions improve the Sinkhorn, KID and FID metrics, which, in turn, translate to corresponding gains in the object detector's performance as well. The improvements in visual quality achieved by higher resolution synthesis are also evident in the first two columns of Fig. 4. We further observed that training the detector without background target images found with Grad-CAM results in false positive detections and reduces mAP from 56.5 to 51.6.

Coupled Training. Next, we evaluate the performance of variants of SSOD trained with coupled synthesis ($\mathcal{S}$)

Method	Coupled	Easy ↑	Medium ↑	Hard ↑	All ↑	Sinkhorn [9] ↓	KID [5] ↓	FID [19] ↓
BlockGAN [37] 64	✗	65.1	48.3	40.5	51.3	0.486	0.048	8.3
BlockGAN [37] 128	✗	69.4	49.9	44.2	54.5	0.483	0.046	7.8
BlockGAN [37] 256	✗	72.7	52.1	44.8	56.5	0.481	0.045	7.61
SSOD *w/o* $\mathcal{L}_{fg} + \mathcal{L}_{bg}$	✓	74.7	59.3	52.7	62.2	0.475	0.042	7.22
SSOD *w/o* $\mathcal{L}_{mso}$	✓	78.3	65.6	53.5	65.8	0.471	0.040	6.86
SSOD *w/o* OSA	✓	76.1	61.3	50.9	62.7	0.475	0.042	7.23
SSOD-Full	✓	**80.8**	**68.1**	**56.6**	**68.4**	**0.465**	**0.037**	**6.37**

Table 1. **Ablation study on KITTI.** Rows 1-3: BlockGAN in $\mathcal{S}$ trained without coupling to the detector at different image resolutions; rows 4-6: different ablated versions of SSOD each with one component removed; and row 7: full SSOD model. Columns 1-3: mAP value at IOU 0.5 for KITTI's Easy, Medium, Hard and All cases; and columns 4-6: Sinkhorn, KID, and FID scores to compare object regions in synthesized and real-world KITTI images.

and object detection ($\mathcal{F}$) networks. We evaluate four variants of SSOD: (a) without the target data appearance adaption losses described in Sec. 3.5 (SSOD *w/o* $\mathcal{L}_{fg} + \mathcal{L}_{bg}$); (b) without the multi-scale object synthesis loss $\mathcal{L}_{mso}$ described in Sec. 3.4 (SSOD *w/o* $\mathcal{L}_{mso}$); (c) without adaptation to the target dataset's object scales as described in Sec. 3.5 (SSOD *w/o* OSA); and (d) the full SSOD model (SSOD-full). We observe that, across the board, all variants of SSOD trained with a coupled detector (bottom four rows of Table 1) perform significantly better than those without (top three rows of Table 1). This result verifies the usefulness of our proposed end-to-end framework, which adapts the synthesis network $\mathcal{S}$ to both the downstream task of object detection as well as to the target dataset's distribution. The best performance, overall, is achieved by our full SSOD model with the highest mAP score of 68.4. Removing each of our individual proposed modules for target data appearance adaptation (SSOD *w/o* $\mathcal{L}_{fg} + \mathcal{L}_{bg}$), target object scale adaptation (SSOD *w/o* OSA) and multi-object scale synthesis (SSOD *w/o* $\mathcal{L}_{mso}$) from SSOD-Full result in a reduction in its performance, with the target data appearance adaption model affecting SSOD's detection accuracy the most.

Qualitative Analysis. We qualitatively evaluate the effect of our proposed losses on the images synthesized by $\mathcal{S}$. In each row of Fig. 4 we show images synthesized with the same foreground and background style codes, but with variants of the network $\mathcal{S}$ trained with a different set of losses in each column. Columns 2-4 are at a resolution of 256×256. We vary the foreground and background style codes across the rows. All objects are synthesized at a large depth from the camera. Fig. 4(a) shows the images synthesized by the original BlockGAN [37] at a resolution of 64×64 suffers from poor quality. Fig. 4(b) shows the synthesized images by our method when trained with the coupled object detector at higher resolution, leads to better visibility. By adding target data appearance adaptation losses ($\mathcal{L}_{fg} + \mathcal{L}_{bg}$), images (Fig. 4(c)) match the appearance of target distribution. Finally, adding the multi-scale object synthesis loss $\mathcal{L}_{mso}$ leads to the best result (high visual quality and appearance alignment to the target distribution). These qualitative results corroborate with their quantitative counterparts: Sinkhorn, KID and FID metrics in Table 2.

Figure 4. **Qualitative analysis of image synthesis.** The columns show images generated by (a) BlockGAN [37] at 64×64; and by $\mathcal{S}$ for (b) SSOD trained without $\mathcal{L}_{fg}$, $\mathcal{L}_{bg}$, and $\mathcal{L}_{mso}$; (c) SSOD trained without $\mathcal{L}_{mso}$; and (d) the full SSOD model. Each row has images generated with the same pose, and foreground and background style codes. Rows (b)-(d) show 256×256 sized images.

4.3. Comparisons to State-of-the-Art

On the KITTI dataset, we compare SSOD to existing methods, Wetectron [42] and PCL [49], capable of training object detectors without requiring bounding box annotations. These methods similar to SSOD, train object detectors solely with unlabeled image collections. They also do not use 3D CAD models and hence are the most directly comparable methods to SSOD. Wetectron [42] is the best-

Method	3D Assets	Easy↑	Medium↑	Hard↑	All↑
PCL [49]	✗	47.3	32.9	19.4	33.2
Wetectron [42]	✗	51.3	37.9	25.1	38.1
SSOD-Full (ours)	✗	**80.8**	**68.1**	**56.6**	**68.4**
Meta-Sim* [26]	✓	65.9	66.3	66.0	66.0
Meta-Sim2 [10]	✓	67.0	67.0	66.2	66.7

Table 2. **Comparisons to SOTA.** Object detection performance (mAP at IOU 0.5) on KITTI of SSOD and various SOTA methods.

performing prior method. We train Wetectron and PCL with a combination of Compcars [52] and KITTI's [15] *training* set; use image-level labels for the presence/absence of the object; get object proposals from Edgeboxes [57]; and evaluate it on KITTI's *validation* set. The results are in Table 2. Compared to Wetectron (mAP of 38.1 for All) and PCL (mAP of 33.2 for All), SSOD (mAP of 68.4 for All) has ∼2X better detection accuracy. We believe that SSOD's superior performance results from its use of a pose-aware synthesizer to generate data for training object detectors. The GAN improves the training data's diversity and also optimally adapts to the task of object detection on target data.

We also compare SSOD to SOTA rendering-based methods Meta-Sim [26] and Meta-Sim2 [10]. They train object detectors purely using synthetically rendered data and evaluate on unlabeled real-world datasets. They require large libraries of 3D CAD models and hence use strong geometric priors. In contrast, SSOD does not use any 3D CAD assets. In fact, its synthesis network can be viewed as a controllable renderer learned only from object image collections without geometric priors. Interestingly, even without using any strong geometric priors, SSOD surpasses both Meta-Sim and Meta-Sim2 for Easy, Medium and All cases in KITTI (Table 2). For Hard cases, SSOD performs lower than Meta-Sim and Meta-Sim2, mostly due its low image quality for occluded objects and its lower 2D bounding box label precision (see Sec. 4.5). Nevertheless, it is exciting that even without using 3D assets and by merely learning from image collections, SSOD can compete with rendering-based methods, which require significant supervision.

4.4. Additional Dataset

An advantage of SSOD is that it can adapt to different target datasets. To validate this, we additionally evaluate it's performance on Cityscapes [8]. We evaluate the full SSOD model trained on Compcars and Cityscapes; its ablated versions with specific individual components removed (as described in Sec. 4.2 – Coupled Training); BlockGAN in $\mathcal{S}$ not coupled with the detector and trained with Compcars only; and the competing Wetectron method trained on Compcars and Cityscapes (Table 3). Similar to KITTI, for Cityscapes too, SSOD-Full achieves the best performance

*We report detection accuracy values for the version of Meta-Sim that does not use labeled validation images from the KITTI [15] dataset.

Method	mAP↑	Sinkhorn↓
Wetectron [42]	18.2	0.549
BlockGAN [37] 256	22.7	0.531
SSOD *w/o* $\mathcal{L}_{fg} + \mathcal{L}_{bg}$	27.2	0.520
SSOD *w/o* $\mathcal{L}_{mso}$	28.5	0.515
SSOD *w/o* OSA	29.1	0.514
SSOD-Full	**31.3**	**0.506**

Table 3. **Performance on Cityscapes.** Object detection performance (mAP at IOU 0.5) and synthetic data quality analysis (Sinkorn) on Cityscapes.

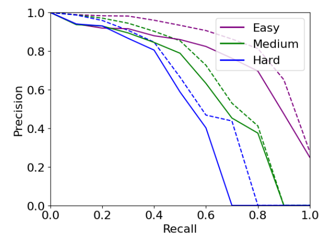

Figure 5. **Precision-recall curves on KITTI.** Curves for SSOD with IOU thresholds of 0.5 (bold lines) and 0.45 (dashed lines).

(mAP of 31.3). Removing $\mathcal{L}_{fg} + \mathcal{L}_{bg}$, which help adapt SSOD to Cityscapes, affects its performance the most. All variants of SSOD jointly trained with the detector perform better than the uncoupled BlockGAN in $\mathcal{S}$. SSOD-Full also performs significantly better than Wetectron (mAP of 18.2).

4.5. Discussion on Results

SSOD suffers from low recall for the Hard cases in KITTI as it fails to detect heavily occluded cars (examples in supplementary material). Fig. 5 shows SSOD's precision-recall curves on KITTI for IOU thresholds: 0.5 (solid) and 0.45 (dashed). Also, with a lower IOU threshold of 0.45 its mAP improves: 80.8 to 83.5 (Easy), 68.1 to 73.2 (Medium) and 56.6 and 63.6 (Hard). This indicates that improving the precision of the synthesized objects' bounding boxes labels can lead to improvements in SSOD's performance.

5. Conclusion

SSOD is the first work to leverage controllable GANs to learn object detectors in a self-supervised manner with unlabelled image collections. It not only opens up an exciting new research paradigm in the area, but also shows that significant detection accuracy can be achieved by using controllable image synthesis. Controllable GANs are able to synthesize data with diversity and realism to train object detectors. They also allow the flexibility to adapt them optimally via end-to-end training to the downstream detection task and target domains. With the rapid progression of controllable GANs, we envision that the gains acquired there would lead to further improvements on GAN-based self-supervised object detection.

References

[1] P. Arbeláez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *CVPR*, 2014. 2

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *ICML*, 2017. 4

[3] Adam Bielski and Paolo Favaro. Emergence of object segmentation in perturbed generative models. In *NeurIPS*, 2019. 2

[4] H. Bilen and A. Vedaldi. Weakly supervised deep detection networks. In *CVPR*, 2016. 2

[5] Mikołaj Bińkowski, Dougal J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying MMD GANs. In *ICLR*, 2018. 6, 7

[6] Yohann Cabon, Naila Murray, and Martin Humenberger. Virtual KITTI 2. *arXiv.org*, 2020. 2

[7] Edo Collins, Radhakrishna Achanta, and Sabine Susstrunk. Deep feature factorization for concept discovery. In *ECCV*, 2018. 2

[8] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 2, 6, 8

[9] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *NeurIPS*, 2013. 6, 7

[10] Jeevan Devaranjan, Amlan Kar, and Sanja Fidler. Meta-sim2: Learning to generate synthetic datasets. In *ECCV*, 2020. 1, 2, 8

[11] Sébastien Ehrhardt, Oliver Groth, Aron Monszpart, Martin Engelcke, Ingmar Posner, Niloy J. Mitra, and Andrea Vedaldi. RELATE: Physically plausible multi-object scene synthesis using structured latent spaces. *NeurIPS*, 2020. 2

[12] Nils Gählert, Nicolas Jourdan, Marius Cordts, Uwe Franke, and Joachim Denzler. Cityscapes 3d: Dataset and benchmark for 9 dof vehicle detection. In *CVPR Workshops*, 2020. 6

[13] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis. In *CVPR*, 2016. 2

[14] Yan Gao, Boxiao Liu, Nan Guo, Xiaochun Ye, Fang Wan, Haihang You, and Dongrui Fan. C-midn: Coupled multiple instance detection network with segmentation guidance for weakly supervised object detection. In *ICCV*, 2019. 2

[15] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *CVPR*, 2012. 1, 2, 6, 8

[16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014. 1

[17] Paul Henderson and Vittorio Ferrari. Learning single-image 3D reconstruction by generative modelling of shape, pose and shading. *IJCV*, 2019. 2

[18] Paul Henderson, Vagia Tsiminaki, and Christoph Lampert. Leveraging 2D data to learn textured 3D mesh generation. In *CVPR*, 2020. 2

[19] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, 2017. 6, 7

[20] Wei-Chih Hung, Varun Jampani, Sifei Liu, Pavlo Molchanov, Ming-Hsuan Yang, and Jan Kautz. Scops: Self-supervised co-part segmentation. In *CVPR*, 2019. 2

[21] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017. 5

[22] Tomas Jakab, Ankush Gupta, Hakan Bilen, and Andrea Vedaldi. Unsupervised learning of object landmarks through conditional image generation. In *NeurIPS*, 2018. 1, 2

[23] Zequn Jie, Yunchao Wei, Xiaojie Jin, Jiashi Feng, and Wei Liu. Deep self-taught learning for weakly supervised object localization. In *CVPR*, 2017. 2

[24] Angjoo Kanazawa, Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *ECCV*, 2018. 2

[25] Vadim Kantorov, Maxime Oquab, Minsu Cho, and Ivan Laptev. Contextlocnet: Context-aware deep network models for weakly supervised localization. In *ECCV*, 2016. 2

[26] Amlan Kar, Aayush Prakash, Ming-Yu Liu, Eric Cameracci, Justin Yuan, Matt Rusiniak, David Acuna, Antonio Torralba, and Sanja Fidler. Meta-sim: Learning to generate synthetic datasets. In *ICCV*, 2019. 1, 2, 8

[27] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *ICLR*, 2018. 4

[28] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019. 4

[29] Nilesh Kulkarni, Abhinav Gupta, and Shubham Tulsiani. Canonical surface mapping via geometric cycle consistency. In *ICCV*, 2019. 2

[30] Xueting Li, Sifei Liu, Shalini De Mello, Kihwan Kim, Xiaolong Wang, Ming-Hsuan Yang, and Jan Kautz. Online adaptation for consistent mesh reconstruction in the wild. In *NeurIPS*, 2020. 2

[31] Xueting Li, Sifei Liu, Kihwan Kim, Shalini De Mello, Varun Jampani, Ming-Hsuan Yang, and Jan Kautz. Self-supervised single-view 3d reconstruction via semantic consistency. In *ECCV*, 2020. 2

[32] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 1, 4

[33] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 2

[34] Siva Karthik Mustikovela, Varun Jampani, Shalini De Mello, Sifei Liu, Umar Iqbal, Carsten Rother, and Jan Kautz. Self-supervised viewpoint learning from image collections. In *CVPR*, 2020. 1, 2

[35] K L Navaneet, Ansu Mathew, Shashank Kashyap, Wei-Chih Hung, Varun Jampani, and R Venkatesh Babu. From image collections to point clouds with self-supervised shape and pose networks. In *CVPR*, 2020. 2

[36] Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. Hologan: Unsupervised learning of 3d representations from natural images. In *ICCV*, 2019. 1, 2

[37] 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, 2, 3, 4, 6, 7, 8

[38] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *CVPR*, 2021. 1, 2

[39] Aayush Prakash, Shaad Boochoon, M. Brophy, David Acuna, Eric Cameracci, Gavriel State, Omer Shapira, and Stan Birchfield. Structured domain randomization: Bridging the reality gap by context-aware synthetic data. *ICRA*, 2019. 2

[40] 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

[41] Zhongzheng Ren, Zhiding Yu, Xiaodong Yang, Ming-Yu Liu, Yong Jae Lee, Alexander G. Schwing, and Jan Kautz. Instance-aware, context-focused, and memory-efficient weakly supervised object detection. In *CVPR*, 2020. 1, 2

[42] Zhongzheng Ren, Zhiding Yu, Xiaodong Yang, Ming-Yu Liu, Alexander G. Schwing, and Jan Kautz. Ufo^2: A unified framework towards omni-supervised object detection. In *ECCV*, 2020. 1, 2, 7, 8

[43] Stephan R. Richter, Zeeshan Hayder, and Vladlen Koltun. Playing for benchmarks. In *ICCV*, 2017. 2

[44] Gernot Riegler and Vladlen Koltun. Stable view synthesis. In *ECCV*, 2021. 2

[45] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. In *NeurIPS*, 2020. 1, 2

[46] 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*, 2017. 5, 6

[47] Yunhan Shen, Rongrong Ji, Shengchuan Zhang, Wangmeng Zuo, and Yan Wang. Generative adversarial learning towards fast weakly supervised detection. In *CVPR*, 2018. 2

[48] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 6

[49] Peng Tang, Xinggang Wang, Song Bai, Wei Shen, Xiang Bai, Wenyu Liu, and Alan Yuille. PCL: Proposal cluster learning for weakly supervised object detection. *TPAMI*, 2018. 2, 7, 8

[50] J. Thewlis, H. Bilen, and A. Vedaldi. Unsupervised learning of object landmarks by factorized spatial embeddings. In *ICCV*, 2017. 1, 2

[51] J.R.R. Uijlings, K.E.A. van de Sande, T. Gevers, and A.W.M. Smeulders. Selective search for object recognition. *IJCV*, 2013. 2

[52] Linjie Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. A large-scale car dataset for fine-grained categorization and verification. In *CVPR*, 2015. 1, 6, 8

[53] Zhaoyang Zeng, Bei Liu, Jianlong Fu, Hongyang Chao, and Lei Zhang. Wsod2: Learning bottom-up and top-down objectness distillation for weakly-supervised object detection. In *ICCV*, 2019. 1, 2

[54] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv.org*, 2020. 2

[55] Xiaopeng Zhang, Jiashi Feng, Hongkai Xiong, and Qi Tian. Zigzag learning for weakly supervised object detection. In *CVPR*, 2018. 2

[56] Yuting Zhang, Yijie Guo, Yixin Jin, Yijun Luo, Zhiyuan He, and Honglak Lee. Unsupervised discovery of object landmarks as structural representations. In *CVPR*, 2018. 1, 2

[57] Larry Zitnick and Piotr Dollar. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014. 1, 2, 8