DISSERTATION

submitted

to the

Combined Faculty of Natural Sciences and Mathematics

of

Heidelberg University, Germany

for the degree of

Doctor of Natural Sciences

Put forward by

M.Sc. Siva Karthik Mustikovela

Born in: Hyderabad, India

Oral examination:

Solving Computer Vision Problems through Self-Supervision and Generative Image Synthesis

Advisor: Prof. Dr. Carsten Rother

Acknowledgments

There are numerous people I would like to acknowledge and I cannot thank them enough for being with me throughout my journey. First of all, I thank my supervisor, Prof. Carsten Rother for giving me an amazing opportunity to pursue my doctoral studies under his supervision. He believed in me throughout, challenged me with new ideas, guided me and provided a lot of academic and industrial opportunities.

Prof. Andreas Geiger also played an influential role during my journey where I was fortunate enough to collaborate with him in several projects. He guided me about approaching research through a clear and organized way, tackling failures and pushing the boundaries of research.

I had the fortune of doing two internships at Nvidia research where I collaborated with Dr. Jan Kautz, Dr. Varun Jampani, Dr. Shalini De Mello, Dr. Umar Iqbal, Dr. Sifei Liu and Aayush Prakash. These internships were the most exciting and enriching parts of my doctoral journey. I would like to thank Dr. Jan Kautz and Dr. Varun Jampani for gracing me with an internship opportunity the first time. Varun is a role model for me who provided exceptional support throughout my internship and even after it ended. He taught me research focus, rigor and exposed me to radical new ideas. During my second internship I was mentored by Dr. Shalini De Mello who has been a very influential mentor in my life. She was extremely supportive in various aspects and it particularly mattered since it was a remote internship during COVID-19 pandemic. She was always available to advise for research, personal struggles and encouraged me to pursue bold ideas.

I had the most amazing company I could have asked for in terms of colleagues or rather friends for life. I cannot imagine myself going though this process without my lab mates Omid, Hassan, Weihao and Sid. They made the struggles enjoyable with super hilarious conversations, helpful and kind acts, emotional support, technical discussions and awesome food. I can never forget the long sleepless nights at the lab where we are sharing the stress about our research while having laughing fits. I could not have asked for better

lab mates. My time at Nvidia was fun filled because of my amazing co-interns, Vinu, Abhishek, Ekta, Adrian, Zahra and Shoaib.

Being thankful or grateful to my parents (Krishna, Aruna) and my sister (Sravani) would be an understatement for what they have gone through, sacrificed and how they supported me during this journey. They thoroughly encouraged me to pursue my dreams. They emotionally supported me when I was feeling low, shared my happiness when I succeeded and inspired me to excel at life. I could not have asked for more from them. I would forever be indebted to them for their support. I also met my amazing girlfriend and now wife, Vinita during my journey and she has been an incredible support system throughout. She constantly motivated me to push my boundaries, encouraged me to take up new challenges, patiently listened to my rants, advised me to deal with my problems and supported me with all my decisions. Most importantly, she believed in my ability and patiently waited for me to go through this journey. I cannot be grateful enough for all our late night phone calls during our long distance relationship which were majorly about us having fun conversations, celebrating out little wins, sharing our struggles, motivating and encouraging each other. I would also like to thank my cousin, Teja who stayed with me through the pandemic making it a little more bearable when we had fun conversations, arguments, pondered about life and made amazing tea. He taught me optimism and hope when met with challenges in life.

Abstract

Computer vision models require large amounts of labeled data for training which is error prone, time consuming and notoriously hard to acquire. It is specifically difficult to obtain labels for fine grained geometry based tasks like object viewpoint estimation and geometry estimation. Obtaining large scale object detection labels for changing operating domains is also time consuming. Synthetic data is an alternative but has a huge domain gap compared to real world images which leads to models to under perform on real images. On the other hand, it is relatively easy to mine large amounts of unlabeled images of an object category from the internet. We seek to answer the whether such unlabeled collections of in-the-wild images can be successfully utilized to train computer vision models purely via self-supervision. We propose methods to learn object viewpoint estimation, object detection, controllable image generation and decomposition purely through self-supervision using unlabeled images in an analysis-by-synthesis paradigm. For object viewpoint estimation, we leverage a viewpoint aware image synthesis network as a form of self-supervision to train our viewpoint estimation network by coupling both the models through cycle-consistency. Our method performs competitively compared to fully supervised methods for objects like faces, cars, busses and trains. For self-supervised object detection, we leverage a generative model which provides control over 3D location and orientation of the synthesized object, using which we also obtain the bounding box of the object. The synthesized image and bounding box are used to train the object detector. The object detection accuracies indicate that we outperform existing baselines considerably and surpass other synthetic data based detection methods. Finally, we propose a method to learn geometrically controlled image generation and decomposition using class specific unpaired real world images and 3D CAD models. We jointly model the forward process of image generation and inverse process of image decomposition. We are able to generate highly realistic images with fine grained control over shape, appearance and reflections. Our results indicate that computer vision tasks can be learned through self-supervision and can achieve performance similar to either supervised methods or synthetic data based methods.

Zusammenfassung

Computer-Vision-Modelle erfordern große Mengen an gelabelten Daten für das Training, was fehleranfällig, zeitaufwändig und notoriously schwer zu beschaffen ist. Es ist besonders schwierig, Labels für feinkörnige geometriebasierte Aufgaben wie die Schätzung der Objektperspektive und die Geometrieschätzung zu erhalten. Das Beschaffen von groß angelegten Labels für die Objekterkennung in sich verändernden Betriebsdomänen ist ebenfalls zeitaufwändig. Synthetische Daten sind eine Alternative, weisen jedoch eine erhebliche Domänenlücke im Vergleich zu echten Weltbildern auf, was dazu führt, dass Modelle auf echten Bildern unterdurchschnittlich abschneiden. Andererseits ist es relativ einfach, große Mengen ungelabelter Bilder einer Objektkategorie aus dem Internet zu gewinnen. Wir versuchen zu beantworten, ob solche ungelabelten Sammlungen von Bildern aus freier Wildbahn erfolgreich genutzt werden können, um Computer-Vision-Modelle ausschließlich über Selbstüberwachung zu trainieren. Wir schlagen Methoden vor, um die Schätzung der Objektperspektive, die Objekterkennung, die steuerbare Bildgenerierung und die Zerlegung ausschließlich durch Selbstüberwachung unter Verwendung ungelabelter Bilder in einem Analyse-durch-Synthese-Paradigma zu erlernen. Für die Schätzung der Objektperspektive nutzen wir ein perspektivbewusstes Bildsynthese-Netzwerk als Form der Selbstüberwachung, um unser Perspektivschätzungsnetzwerk zu trainieren, indem wir beide Modelle durch Zykluskonsistenz koppeln. Unsere Methode ist wettbewerbsfähig im Vergleich zu vollständig überwachten Methoden für Objekte wie Gesichter, Autos, Busse und Züge. Für die selbstüberwachte Objekterkennung nutzen wir ein generatives Modell, das die Kontrolle über den 3D-Standort und die Ausrichtung des synthetisierten Objekts bietet, mit dem wir auch den Begrenzungsrahmen des Objekts erhalten. Das synthetisierte Bild und der Begrenzungsrahmen werden zur Schulung des Objekterkenners verwendet. Die Genauigkeit der Objekterkennung zeigt, dass wir vorhandene Baselines erheblich übertreffen und andere auf synthetischen Daten basierende Erkennungsmethoden überbieten. Schließlich schlagen wir eine Methode vor, um die geometrisch gesteuerte Bildgenerierung und Zerlegung

unter Verwendung klassenspezifischer ungleichartiger realer Weltbilder und 3D-CAD-Modelle zu erlernen. Wir modellieren gemeinsam den Vorwärtsprozess der Bildgenerierung und den inversen Prozess der Bildzerlegung. Wir können äußerst realistische Bilder mit feinkörniger Kontrolle über Form, Erscheinungsbild und Reflexionen generieren. Unsere Ergebnisse deuten darauf hin, dass Computer-Vision-Aufgaben durch Selbstüberwachung erlernt werden können und eine Leistung erreichen können, die der von überwachten Methoden oder auf synthetischen Daten basierenden Methoden ähnelt.

Contents

List of Figures

List of Tables

“One day, in retrospect, the years of struggle will strike you as the most beautiful.”

Sigmund Freud

Chapter 1

Introduction

Computer Vision has progressed extensively with the development of several methods that can understand a given scene at various levels of granularity. Current day models are able to achieve unprecedented accuracy at scene understanding tasks like image classification, semantic and instance segmentation, object detection and object pose estimation to name a few. Simultaneously, there have also been significant advances in generative models which are able to synthesize high quality realistic images which are able to trick humans into thinking they are real images. Convolutional Neural Networks played an integral role in building advanced computer vision algorithms for scene understanding tasks. CNNs are trained with large amounts of labeled data to achieve impressive scene understanding performance. The efforts to build large amount of high quality labeled datasets enabled the training of such complex data hungry algorithms. In a standard supervised setting, CNNs are trained using a large amount of paired labeled data in the form of $(input, label)$. The $input$ could be from sensor modalities like 2D RGB images, 3D point clouds, audio, text, etc. The $label$ provides a target that the model should predict depending on the task it is being trained for.

1.1 Supervised Learning in Computer Vision

Semantic Understanding. There are several common computer vision tasks to provide a high-level semantic understanding of a given scene and the objects, namely, image classification, semantic segmentation, object detection, instance segmentation (Fig 1.1). Image classification is the task of categorizing an image into a predefined set of classes requiring a simple categorical $label$ per image as target annotation. Semantic segmentation is a more granular task that classifies each pixel in the image and hence requires dense pixel-wise class $label$ target annotations. Furthermore, object detection is a task that localizes the objects present in a scene by predicting their bounding boxes and simultaneously assigning a class to each of the predicted objects. Instance segmentation takes this one step further by predicting a dense pixel level mask of each detected object in the scene. Object detection and instance segmentation models require target annotations in the form of individual bounding box and fine-grained masks for each of the object in the scene. Object detection is a fundamental task in computer vision that enables machines to interpret and interact with the visual world. For e.g. object detection is essential for self-driving cars and other autonomous vehicles to recognize and track pedestrians, other vehicles, traffic signs, and obstacles in real-time, ensuring safe navigation.

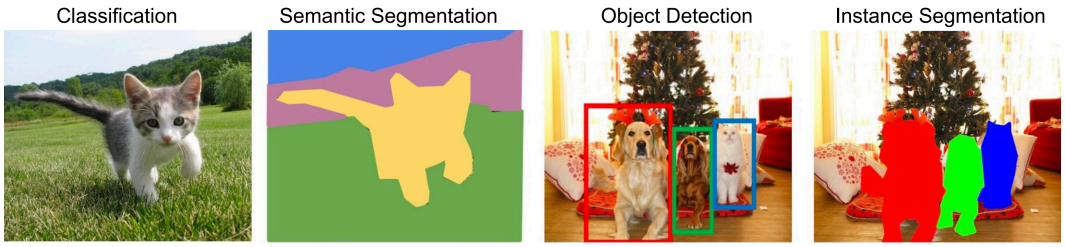


Figure 1.1. Semantic understanding tasks in computer vision. (a) Image classification predicts an object category per image. (b) Semantic segmentation is more fine grained and predicts an object category per pixel. (c) Object detection predicts a bounding box for each object in the image. (d) Instance segmentation further predicts a pixel level object mask. Image from [1]

Geometric Understanding. Computer vision tasks like pose estimation, object reconstruction, depth estimation, optical flow estimation, scene flow estimation, etc., provide

a geometric understanding of the scene. Object pose estimation and 3D object detection models recover the three dimensional location (x, y, z) and three dimensional orientation $(azimuth, elevation, tilt)$ of an object requiring fine grained continuous valued six dimensional annotations (Fig 1.2). Object reconstruction models predict the fine geometric shape of an object allowing us to recover a computerized model of the object requiring object level six dimensional pose annotations along with a corresponding 3D computerized model of the object. Both object pose estimation and reconstruction are vital tasks for applications like autonomous driving, augmented reality, robotic manipulation, etc. At the scene level, depth estimation (Fig 1.3(c)) recovers a pixel wise depth value indicating the distance of that point from the camera in 3D space. Optical flow (Fig 1.3(d)) tracks all pixels in a scene and associates them with the pixels in the next temporal scene. While depth estimation models require pixel wise depth annotations in camera frame, optical flow models need pixel wise tracking ground truth indicating pixel wise associations across temporal frames.

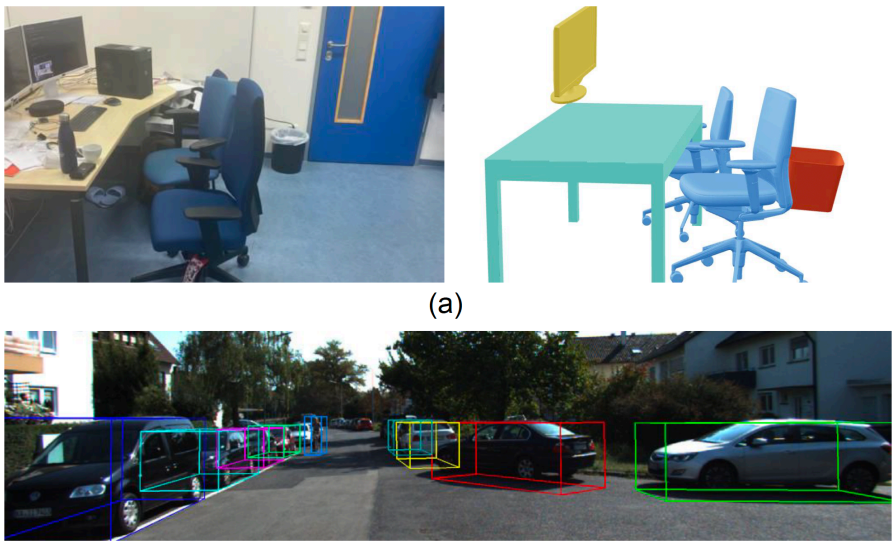


Figure 1.2. (a) Example of 6D object pose estimation in indoor scenes from Scan2CAD dataset [8]. Chairs, table and computer monitor are aligned and annotated with the corresponding 3D CAD models. Image from [62]. (b) Example of outdoor 3D object detection labels from KITTI dataset [49].

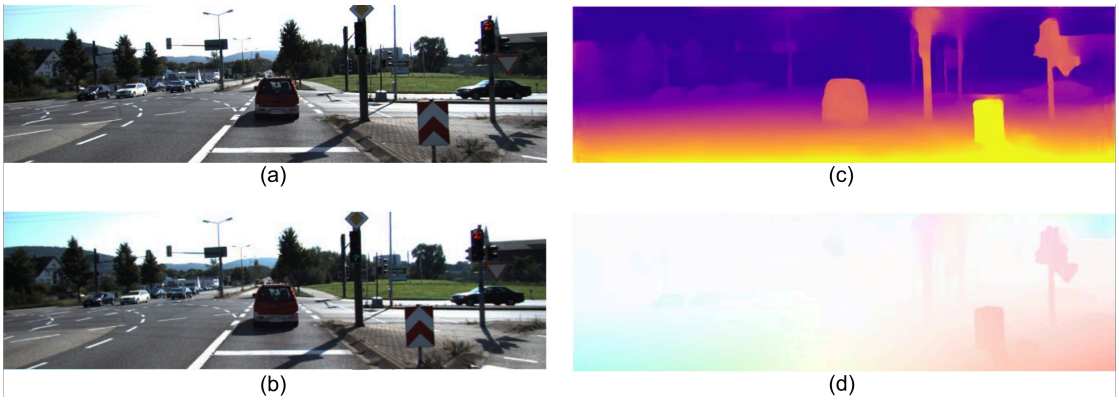


Figure 1.3. Geometric scene understanding tasks in computer vision. (a) Image at time t (b) Image at time t+1 (c) Pixel wise depth prediction of the scene (d) Pixel wise optical flow prediction of the scene.

1.2 Challenges in data acquisition

Semantic Scene understanding. High-level semantic understanding tasks need pixel wise masks and corresponding categorical class labels which require extensive human effort and are also expensive to obtain. Works like [33] (Cityscapes), [129] (Mapillary) provide densely annotated pixel wise semantic and instance annotations for RGB images in street scenes for autonomous scene understanding applications (Fig 1.4). To create these datasets, it takes about 90 minutes to annotate each image with semantic and instances masks, which is extremely time consuming.

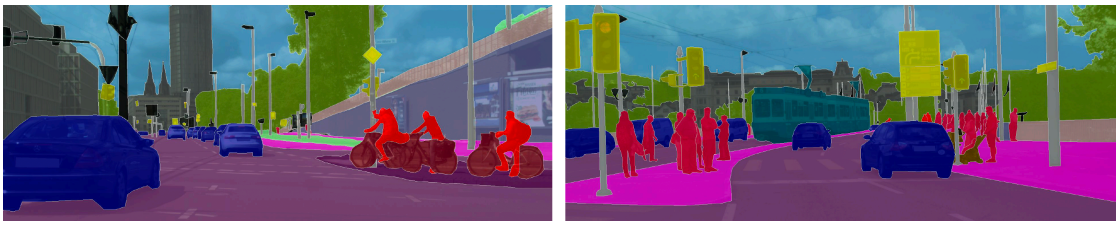


Figure 1.4. Example from Cityscapes outdoor scene understanding dataset. Each image is overlaid with the corresponding pixel wise dense category labels.

3D object detection. 3D bounding box annotations or object pose annotations present the additional challenge of manually estimating the rotation and translation of an object with respect to a canonical frame to accurately place the bounding box covering the ob-

ject. This could be highly ambiguous for humans and gets erroneous when the objects are occluded or farther away from the camera (Fig 1.2(b)). Datasets like PASCAL-3D+[183] provide an annotated rotation and location of an object along with a corresponding CAD model associated with the object. This is extremely challenging since we need a 3D CAD model that exactly matches with the object in consideration and further assign an exact location and pose which is highly ambiguous (Fig 1.2(a)). SUN RGB-D dataset [168] provides 3D bounding boxes for objects present in an indoor scene captured with 3D sensors. It is reported that about 2,051 hours were required to obtain bounding boxes for about 65k 3D object annotations averaging about 2 minutes per object bounding box. Assuming about 15 objects per scene it would take about 30 minutes to annotate a scene.

An advantage in the above mentioned tasks like semantic segmentation and object detection is the possibility of perception of ground truth by humans for verification and quality assessment. For e.g., semantic or instance annotations, 3D or 2D bounding boxes can be overlaid on images to verify if they rightly fit the object. On the other hand, geometric scene understanding tasks (Fig 1.3) usually require annotations that are multi-dimensional, continuous valued and associative. Scene depth estimation requires pixel wise depth annotations which is extremely difficult and near impossible for humans to annotate at fine pixel level. Depth annotations are generally obtained by simultaneously capturing RGB images and the corresponding LiDAR scans at the same time followed by calibrating and registering the images and scans [49], [47]. This allows us to capture accurate pixel wise depth for outdoor scenes which however is sparse due to the sparsity of LiDAR scans. For indoor scenes, depth for corresponding RGB images is obtained through time-of-flight cameras like Kinect. Optical flow on the other hand requires pixels to be tracked across time through pairs of images. It is practically impossible for humans to annotate for optical flow and pixel level association across time. To obtain optical flow ground truth, images and LiDAR scans are registered across time and the 3D points from LiDAR scans projected into the next frame to obtain pixel level correspondences [49]. This is a very complicated approach and does not scale well when there are multiple dynamic objects in a scene.

Synthetic data. One possibility of addressing the above mentioned challenges in acquir-

ing real data is to leverage computer graphics to generate synthetic datasets [45], [157], [155]. Computer graphics pipelines allow us easy access to various kinds of high quality pixel-precise annotations like segmentation, bounding boxes, depth, optical flow, object pose, etc. This makes it easier to create high-quality labeled datasets for training. Synthetic data generation also allows us to precisely control the composition of the scene according to our need enabling us to capture highly diverse scenarios that might be rare or difficult to capture in the real world. However, there are a few major shortcomings associated with synthetic data. Synthetic datasets suffer from domain gap due to inherent differences between synthetic and real data distributions and hence models trained solely on synthetic data may not always generalize well to real-world scenarios. Additionally, synthetic datasets require skilled artists who can create assets by modeling real world objects, compose scenes in graphics pipelines and create high quality near real world imagery which in itself is a heavily studied problem in computer graphics. To this end, synthetic data can only be used in combination with real data and cannot entirely replace it to train computer vision models.

As it can be seen in the above mentioned examples, obtaining ground truth for real images is extremely expensive, cumbersome and in some cases untractable. On the other hand, building synthetic datasets poses challenges due to the domain gap and expertise required to build them. The goal of this thesis is to explore methods which let us learn computer vision models without using annotated training data either from real or synthetic setting, merely by using large collections of images. We explore how fundamental problems in computer vision like pose estimation, object detection, image generation and image decomposition can be solved by training models through the paradigm of analysis-by-synthesis and self-supervision without the need of annotated training data.

1.3 Self-Supervision in Computer Vision

Self-supervised learning has shown remarkable success in various computer vision tasks, including image classification, object detection, and segmentation. It has the advantage of being able to learn meaningful representations from large, unlabeled datasets, which

is especially valuable when labeled data is scarce or expensive to obtain. Recently, self-supervised methods like SimCLR [28], MoCo [63], BYOL [59], PIRL [123], Dino [23] have demonstrated tremendous success at learning powerful representations for image classification with needing extensive labeled data. Such methods only require a very minute amount of labeled data to align the learned representations to the object categories. These methods generally learn representations through a pretext task like image colorization, image augmentation, contrastive learning, sorting a sequence of frames, inpainting images, etc. Features from an intermediate layer are then fed to an MLP classifier for ImageNet classification. On the other hand, there has also been significant progress in self-supervised learning of other computer vision tasks. Methods like [53], [54], [52], [61], [148] use stereo imagery or sequential frames to learn depth estimation of outdoor driving scenes. These methods learn by predicting depth and warping it to either the paired stereo frame or the next temporal frame. The resulting warped image is then compared to reference stereo or temporal image to compute loss and penalize the model. Similarly, [83], [115] learn optical flow computation in a self-supervised setup only by using videos. Self-supervised object reconstruction is proposed in [86], [110] which can recover 3D textured object meshed from 2D images. SCOPS [75] and [80] propose methods to learn object part segmentation and keypoint detection only using unlabeled 2D images. All of the above mentioned methods bypass the need for labeled data and solely learn from unlabeled 2D images. These are particularly impactful since obtaining ground truth for the above tasks is extremely expensive and sometimes intractable as mentioned in 1.2. In this thesis, we further explore other computer vision tasks like object pose estimation, object detection, image generation and decomposition which can be learned through self-supervision only using image collections.

1.4 Generative Adversarial Networks

Generative Adversarial Networks (GANs) [55] are a common and powerful class of machine learning models which have successfully demonstrated the ability to model the distribution of natural images and further generate realistic looking image samples.

Vanilla unconditional models like [91], [19], [92] are able to generate high resolution images very similar to the real images they are trained on. GANs consist of two networks, a Generator ($\mathcal{G}$) and a discriminator ($\mathcal{D}$) during training. In the vanilla setup, the generator $\mathcal{G}$ takes a vector z sampled from a predetermined distribution as input and produces an image $I' = \mathcal{G}(z)$. The discriminator, $\mathcal{D}$ is another network which either takes a real image I or fake (generated) image I' and identifies whether the image presented to it is synthesized by $\mathcal{G}$ (from fake distribution) or real distribution. While vanilla GAN models are able to generate highly realistic images compared to natural images (Fig 1.5(a)), the generated images are not controllable. To address this, several methods have been proposed to condition the generator on various inputs like object category, style, text, semantic composition, object pose, object location, etc. Image translation GANs like [200], [77] were one of the first models to introduce conditional GANs where the input to $\mathcal{G}$ is not just a random vector but an alternative representation or a map of the image we want from the generator (Fig 1.5(b,c)). Such conditional GANs are tasked to translate an image provided to $\mathcal{G}$ into another image. For e.g. $\mathcal{G}$ could translate a semantic map into its corresponding image, a gray scale image into a color image, a binary edge image into its color image,etc. Models like [179], [140] further improved the performance of semantic map condition based image generation by achieving high level of realism of synthesized images (Fig 1.5(b)). [193] demonstrated that $\mathcal{G}$ could be conditioned on textual inputs to control the characteristics of generated images. At a more fine grained level, [130] showed the possibility of having fine grained control over the pose of generated image using pose conditioned generator (Fig 1.5(d)). On the other hand [133] allowed control over the location of an object in the synthesized scene. While the above mentioned works are a few methods of conditioning GANs, there are numerous methods of controlling the generated images developed over the past few years, some of which are discussed in [88]. A key takeaway from the above methods is the possibility of controlling the high level or fine-grained attributes of scenes and objects in the generated images. An efficient generator is able to take a parametric control p as input and faithfully incorporate the required characteristics into the generated image $I' = \mathcal{G}(p)$. The synthesized image can in turn be analyzed by a downstream task network to predict the parameters used to produce the image. This effectively becomes a self learning loop

where (I', p) can be used as training samples and is one of the major building blocks of this thesis.

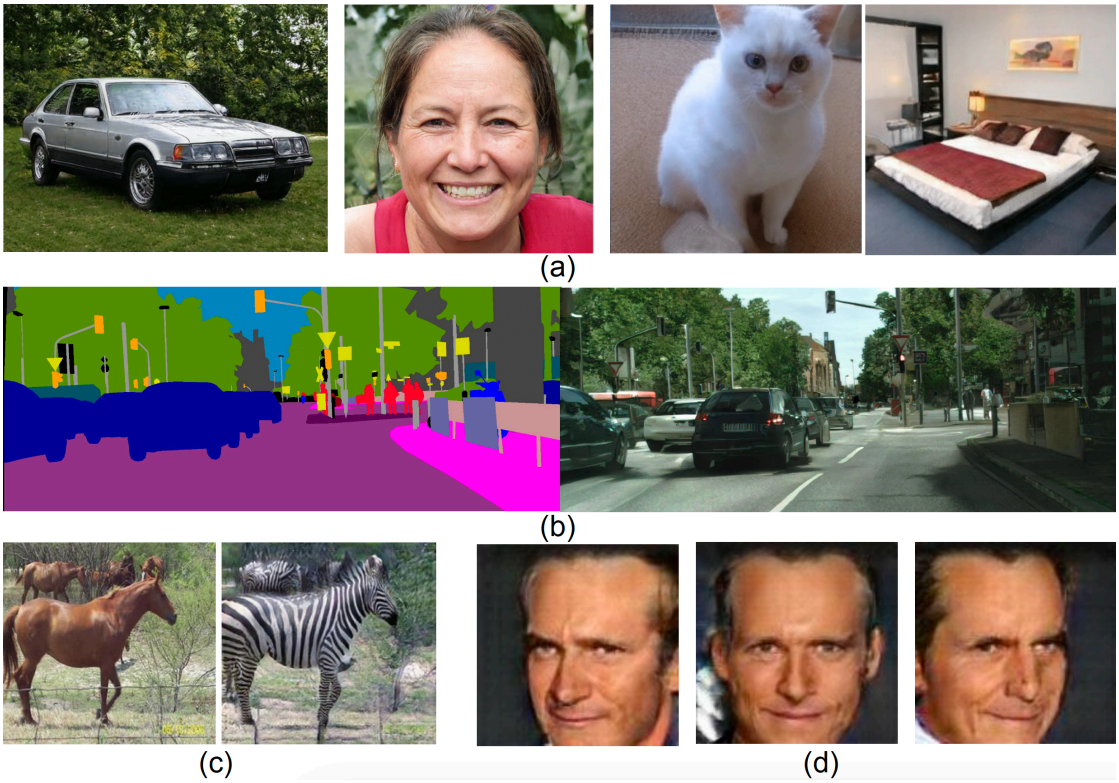


Figure 1.5. Examples produced by various Generative Adversarial Networks. (a) Image samples from StyleGAN[92]. (b) Conditional image generation results from Pix2PixHD[179] which uses semantic maps as input to generate RGB images. (c) Example from CUT[139] which can translate images between domains. (d) Images generated from viewpoint controlled generative model HoloGAN[130].

1.5 Motivation

The motivation for this thesis originates from two major factors. The first comes from the challenges involved in data acquisition as described in sec 1.2. While data collection is plausible for some tasks like object detection, it is expensive both in terms of time and human effort. On the other hand, data collection for tasks like pose estimation, geometry estimation, etc is ambiguous, error prone and some times untractable through human

means. Secondly, parametrically controllable GANs discussed in 1.4 enable us to steer the characteristics required like pose, style, geometry, etc., in a generated image. Controllable generative models can effectively be used to create $(label, image)$ pairs (I', p) where p is the input to synthesis network and I' is the synthesized image. This pair can further be used to train the down stream computer vision task where I' serves as the input image and p serves as the label. This forms an analysis-by-synthesis cycle where the synthesis network is coupled with the analysis (task) network to form a self-supervised learning system which only required unlabeled real world images as training data. The goal of this thesis is to leverage this controllable image generation in an analysis-by-synthesis paradigm to train computer vision models only by using large collections of unlabeled images. Through this thesis, we explore self-supervised learning of three major computer vision problems, namely, object viewpoint estimation, object detection and image decomposition. We learn each of these tasks through self-supervision in an analysis-by-synthesis framework through losses and constraints tailored to each of these specific problems. Below is the summary of the main contributions of this thesis.

1.6 Contributions

To summarize, the main contributions of thesis are:

- An analysis-by-synthesis framework for learning viewpoint estimation in a purely self-supervised manner by leveraging cycle-consistency losses between a viewpoint estimation and a viewpoint aware synthesis network. Self-supervision here refers to the fact that the only true supervisory signal that the network has is the input image itself. We couple a viewpoint network with a synthesis network to form a complete cycle and train both together. To self-supervise viewpoint estimation, we leverage cycle-consistency losses between the viewpoint estimation (analysis) network and a viewpoint aware generative (synthesis) network, along with losses for viewpoint and appearance disentanglement, and object-specific symmetry priors.We introduce generative, symmetric and adversarial constraints which self-supervise viewpoint estimation learning just from object image col-

lections. We perform experiments for head pose estimation on the BIWI dataset [41] and for viewpoint estimation of cars, buses and trains on the challenging Pascal3D+ [183] dataset and demonstrate competitive accuracy in comparison to fully-supervised approaches. As per our knowledge, this is one of first works to explore the problem of self-supervised viewpoint learning for general objects and we believe that our work would serve as a robust baseline for future work.

- A self-supervised object detection framework SSOD via controllable generative synthesis, which uses only real world images without any kind of bounding box annotations. We use unlabeled image collections to learn to synthesize objects in a controlled way with pre defined object properties. We achieve this by leveraging controllable GANs, which provide control over the 3D location and orientation of an object. Using the synthesized images and the bounding boxes obtained using the corresponding input object properties we train the object detection network. We propose an end-to-end analysis-by-synthesis framework, which can optimally adapt the synthesis network to both the downstream task of object detection and to a target dataset in a purely self-supervised manner without requiring bounding box-labels and without using 3D CAD assets – a multi-faceted challenge not addressed previously. Our experiments on two real-world datasets KITTI [47] and Cityscapes [34] show $\sim$2x performance improvement over SOTA image-based self-supervised object detection methods. To our understanding, this is the first work that explores the use of controllable GANs to learn object detection only using image collections.Also, without using 3D CAD assets, SSOD outperforms on average, the rendering-based baseline of Meta-Sim2 [37] which uses graphics engines to produce synthetic data and train object detectors.

- We propose the Intrinsic Autoencoder, a method to jointly train photo-realistic image synthesis and intrinsic image decomposition using cycle consistency losses without using any paired annotated data. While several supervised methods have been proposed for this task, acquiring a dataset of images with accurately aligned 3D models is very difficult. The main contribution of this work is to lift this restriction by training a neural rendering algorithm from unpaired data. We propose

a model for joint generation of realistic images from synthetic 3D models while simultaneously decomposing real images into their intrinsic shape and appearance properties. In contrast to a traditional graphics pipeline, our approach does not require to specify all scene properties, such as material parameters and lighting manually. Instead, we learn photo-realistic deferred rendering from a small set of 3D models and a larger collection of unaligned real images, both of which are easy to acquire in practice. We propose a shared discriminator network that enables better generalization and proves key for learning both tasks without paired training data. Our experiments confirm that a joint treatment of rendering and decomposition is indeed beneficial and that our approach outperforms state-of-the-art image-to-image translation baselines both qualitatively and quantitatively.

1.7 List of published research papers

The remaining chapters of this thesis are based on the following published research papers.

- **Mustikovela, S. K.**, Jampani, V., De Mello, S., Liu, S., Iqbal, U., Rother, C., & Kautz, J., *Self-Supervised Viewpoint Learning From Image Collections.* In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR) 2020

- **Mustikovela, S. K.**, and De Mello, S., Prakash, A., Iqbal, U., Liu, S., Nguyen-Phuoc T., Rother, C., & Kautz, J., *Self-Supervised Object Detection via Generative Image Synthesis.* In Proceedings of the International Conference on Computer Vision (ICCV) 2021

- **Mustikovela, S. K.***, Alhaija, H. A.*, Thies, J., Jampani, V., Nießner, M., Geiger, A., & Rother, C. *Intrinsic Autoencoders for Joint Deferred Neural Rendering and Intrinsic Image Decomposition.* In International Conference on 3D Vision (3DV) 2020 [5].
  (*: Equal contributions)

Declaration: The idea for this paper and the initial implementation was done by Hassan Abu Alhaija. I led the first stage of experiments and development while Hassan was doing an internship. After returning, Hassan led the second stage of experiments. The paper writing has been jointly written with him.

Additionally, I contributed to the following related research papers during working on this thesis. They will not be discussed here.

- **Karthik Mustikovela, S.***, Behl, A.*, Hosseini Jafari, O.*, Alhaija, H. A., Rother, C., & Geiger, A. *Bounding boxes, segmentations and object coordinates: How important is recognition for 3d scene flow estimation in autonomous driving scenarios?* In Proceedings of the International Conference on Computer Vision (ICCV) 2017 [10].

- **Mustikovela, S. K.*** , Hosseini Jafari, O.*,, Pertsch, K., Brachmann, E., & Rother, C. *iPose: Instance-Aware 6D Pose Estimation of Partly Occluded Objects*. In Asian Conference on Computer Vision (ACCV) 2018 [2].

- Alhaija, H. A., **Mustikovela, S. K.**, Mescheder, L., Geiger, A., & Rother, C. *Augmented reality meets deep learning for car instance segmentation in urban scenes.* In British machine vision conference (BMVC) 2017 [4].

- Alhaija, H. A., **Mustikovela, S. K.**, Mescheder, L., Geiger, A., & Rother, C. *Augmented reality meets computer vision: Efficient data generation for urban driving scenes.* International Journal of Computer Vision (IJCV), 2018 [3].

- Alhaija, H. A., **Mustikovela, S. K.**, Geiger, A., & Rother, C. *Geometric image synthesis.* In Asian Conference on Computer Vision (ACCV) 2018 [2].

- **Mustikovela, S. K.**, Yang, M., & Rother, C. *Can Ground Truth Label Propagation from Video help Semantic Segmentation?* In Video Segmentation Workshop, European Conference on Computer Vision (ECCV) 2016.

1.8 Thesis outline

The remaining part of the thesis is structured as follows. In Chapter 2 we explore and present the topic of self-supervised viewpoint learning of objects by using unlabeled image collections. In Chapter 3 we discuss a method to learn object detection in a self-supervised manner through generative image synthesis. Chapter 4 discusses our approach to jointly learn controllable image synthesis and intrinsic decomposition using unpaired natural images and 3D CAD models. Finally, in Chapter 5 we conclude with a discussion and possible future directions following this thesis.

Chapter 2

Learning Viewpoint Estimation Through Self-Supervision

2.1 Introduction

3D understanding of objects from 2D images is a fundamental computer vision problem. Object viewpoint (azimuth, elevation and tilt angles) estimation provides a pivotal link between 2D imagery and the corresponding 3D geometric understanding. In this work, we tackle the problem of object viewpoint estimation from a single image. Given its central role in 3D geometric understanding, viewpoint estimation is useful in several vision tasks such as object manipulation [181], 3D reconstruction [103], image synthesis [30] to name a few. Estimating viewpoint from a single image is highly challenging due to the inherent ambiguity of 3D understanding from a 2D image. Learning-based approaches, *e.g.*, [58, 60, 112, 120, 169, 177, 186, 199], using neural networks that leverage a large amount of annotated training data, have demonstrated impressive viewpoint estimation accuracy. A key requirement for such approaches is the availability of large-scale human annotated datasets, which is very difficult to obtain. A standard way to annotate viewpoints is by manually finding and aligning a rough morphable 3D or CAD model to images [41, 183, 202], which is a tedious and slow process. This makes it challenging to create large-scale datasets with viewpoint annotations. Most existing

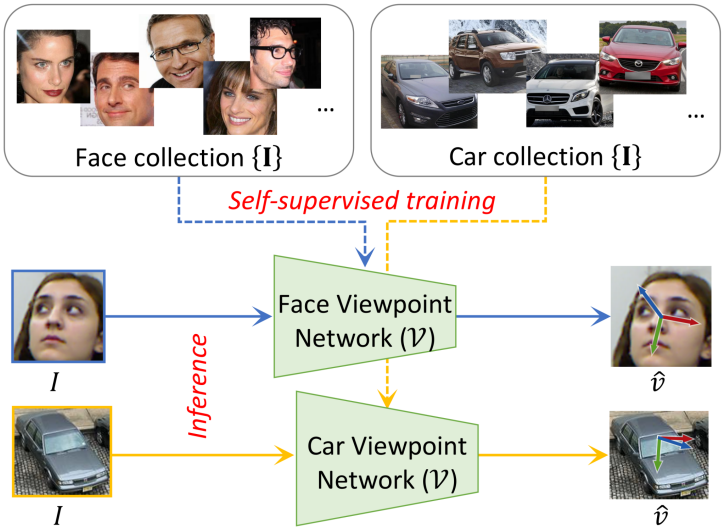


Figure 2.1. Self-supervised viewpoint learning. We learn a single-image object viewpoint estimation network for each category (face or car) using only a collection of images without ground truth.

works [43, 58, 60, 112, 169, 202] either rely on human-annotated viewpoints or augment real-world data with synthetic data. Some works [58] also leverage CAD models during viewpoint inference.

In this work, we propose a self-supervised learning technique for viewpoint estimation of general objects that learns from an object image collection without the need for any viewpoint annotations (Figure 3.1). By image collection, we mean a set of images containing objects of a category of interest (say, faces or cars). Since viewpoint estimation assumes known object bounding boxes, we also assume that the image collection consists of tightly bounded object images. Being self-supervised in nature, our approach provides an important advancement in viewpoint estimation as it alleviates the need for costly viewpoint annotations. It also enables viewpoint learning on object categories that do not have any existing ground-truth annotations.

Following the analysis-by-synthesis paradigm, we leverage a viewpoint aware image synthesis network as a form of self-supervision to train our viewpoint estimation network. We couple the viewpoint network with the synthesis network to form a complete cycle and train both together. To self-supervise viewpoint estimation, we leverage cycle-

consistency losses between the viewpoint estimation (analysis) network and a viewpoint aware generative (synthesis) network, along with losses for viewpoint and appearance disentanglement, and object-specific symmetry priors. During inference, we only need the viewpoint estimation network, without the synthesis network, making viewpoint inference simple and fast for practical purposes. As per our knowledge, ours is the first self-supervised viewpoint learning framework that learns 3D viewpoint of general objects from image collections in-the-wild. We empirically validate our approach on the human head pose estimation task, which on its own has attracted considerable attention [20, 25, 60, 102, 170, 186, 187, 202] in computer vision research. We demonstrate that the results obtained by our self-supervised technique are comparable to those of fully-supervised approaches. In addition, we also demonstrate significant performance improvements when compared to viewpoints estimated with self-supervisedly learned keypoint predictors. To showcase the generalization of our technique, we analyzed our approach on object classes such as cars, buses, and trains from the challenging Pascal3D+ [183] dataset. We believe this work opens up further research in self-supervised viewpoint learning and would also serve as a robust baseline for future work.

To summarize, our main contributions are:

- We propose a novel analysis-by-synthesis framework for learning viewpoint estimation in a purely self-supervised manner by leveraging cycle-consistency losses between a viewpoint estimation and a viewpoint aware synthesis network. To our understanding, this is one of first works to explore the problem of self-supervised viewpoint learning for general objects.

- We introduce generative, symmetric and adversarial constraints which self-supervise viewpoint estimation learning just from object image collections.

- We perform experiments for head pose estimation on the BIWI dataset [41] and for viewpoint estimation of cars, buses and trains on the challenging Pascal3D+ [183] dataset and demonstrate competitive accuracy in comparison to fully-supervised approaches.

2.2 Related Work

Viewpoint estimation Several successful learning-based viewpoint estimation techniques have been developed for general object categories that either regress orientation directly [112, 120, 125, 147, 169, 177]; locate 2D keypoints and fit them to 3D keypoints [58, 145, 199]; or predict 3D shape and viewpoint parameters [103]. These techniques require object viewpoint annotations during training, either in the form of angular values; or 2D and 3D keypoints and use large annotated datasets, *e.g.*, Pascal3D+ [183] and ObjectNet3D [182] with 12 and 100 categories, respectively. These datasets were annotated via a tedious manual process of aligning best-matched 3D models to images – a procedure that is not scalable easily to larger numbers of images or categories. To circumvent this problem, existing viewpoint algorithms augment real-world data with synthetic images [43, 58, 112, 169]; assume auxiliary supervision and learn the related aspects (*e.g.*, 3D keypoints) along with viewpoint [171, 199]; or try to learn from very few labeled examples of novel categories [176].

Head pose estimation Separate from the above-mentioned works, learning-based head pose estimation techniques have also been studied extensively [20, 25, 60, 102, 170, 186, 187, 202]. These works learn to either predict facial landmarks from data with varying levels of supervision ranging from full [20, 102, 170, 187, 202], partial [70], or no supervision [75, 198]; or learn to regress head orientation directly in a fully-supervised manner [25, 60, 158, 186]. The latter methods perform better than those that predict facial points [186]. To avoid manual annotation of head pose, prior works also use synthetic datasets [60, 202]. On the other hand, several works [42, 160, 173, 175] propose learning-based approaches for dense 3D reconstruction of faces via in-the-wild image collections and some use analysis-by-synthesis [173, 175]. However, they are not purely self-supervised and use either facial landmarks [173], dense 3D surfaces [42] or both [175] as supervision.

Self-supervised object attribute discovery Several recent works try to discover 2D object attributes like landmarks [79, 174, 198] and part segmentation [32, 75] in a self-supervised manner. These works are orthogonal to ours as we estimate 3D viewpoint.

Some other works such as [64, 76, 104] make use of differentiable rendering frameworks to learn 3D shape and/or camera viewpoint from a single or multi-view image collections. Because of heavy reliance on differentiable rendering, these works mainly operate on synthetic images. In contrast, our approach can learn viewpoints from image collections in the wild. Some works learn 3D reconstruction from in-the-wild image collections, but use annotated object silhouettes along with other annotations such as 2D semantic keypoints [87], category-level 3D templates [100]; or multiple views of each object instance [94, 135, 180]. In contrast, we use no additional supervision other than the image collections that comprise of independent object images. To the best we know, no prior works propose to learn viewpoint of general objects in a purely self-supervised manner from in-the-wild image collections.

2.3 Self-Supervised Viewpoint Learning

Problem setup We learn a viewpoint estimation network $\mathcal{V}$ using an in-the-wild image collection $\{\mathbf{I}\}$ of a specific object category without annotations. Since viewpoint estimation assumes tightly cropped object images, we also assume that our image collection is composed of cropped object images. Figure 3.1 shows some samples in the face and car image collections. During inference, the viewpoint network $\mathcal{V}$ takes a single object image I as input and predicts the object 3D viewpoint $\hat{\boldsymbol{v}}$.

Viewpoint representation To represent an object viewpoint $\hat{\boldsymbol{v}}$, we use three Euler angles, namely azimuth ($\hat{a}$), elevation ($\hat{e}$) and in-plane rotation ($\hat{t}$) describing the rotations around fixed 3D axes. For the ease of viewpoint regression, we represent each Euler angle, e.g., $a \in [0, 2\pi]$, as a point on a unit circle with 2D coordinates $(\cos(a), \sin(a))$. Following [112], instead of predicting co-ordinates on a 360° circle, we predict a positive unit vector in the first quadrant with $|\hat{a}| = (|\cos(\hat{a})|, |\sin(\hat{a})|)$ and also the category of the combination of signs of $\sin(\hat{a})$ and $\cos(\hat{a})$ indicated by $\text{sign}(\hat{a}) = (\text{sign}(\cos(\hat{a})), \text{sign}(\sin(\hat{a})) \in \{(+,+),(+,-),(-,+),(-,-)\}$. Given the predicted $|\hat{a}|$ and $\text{sign}(\hat{a})$ from the viewpoint network, we can construct $cos(\hat{a}) = \text{sign}(\cos(\hat{a}))|\cos(\hat{a})|$ and $\sin(\hat{a}) = \text{sign}(\sin(\hat{a}))|\sin(\hat{a})|$. The predicted Euler angle $\hat{a}$ can finally be computed as $\tanh(\sin(\hat{a})/\cos(\hat{a}))$. In short,

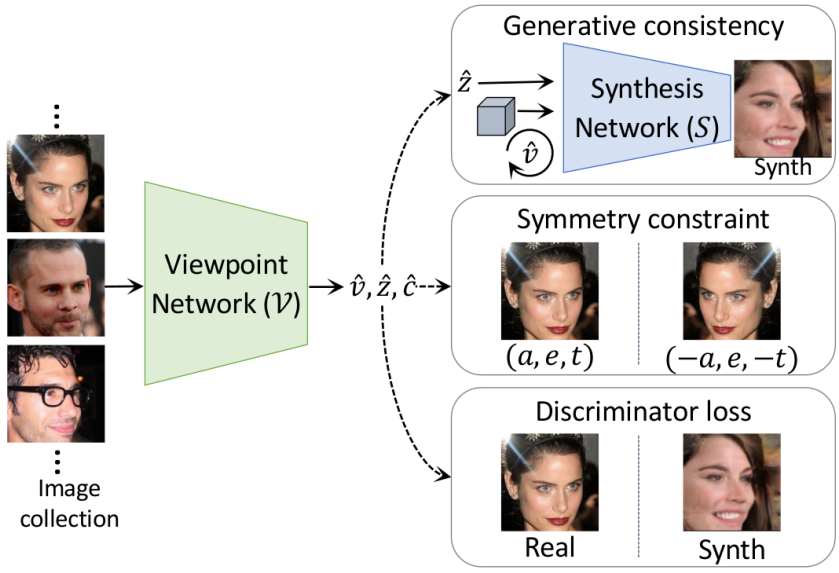


Figure 2.2. Approach overview. We use generative consistency, symmetry and discriminator losses to supervise the viewpoint network with a collection of images without annotations.

the viewpoint network performs both regression to predict a positive unit vector $|a|$ and also classification to predict the probability of $\text{sign}(a)$.

Approach overview and motivation We learn the viewpoint network $\mathcal{V}$ using a set of self-supervised losses as illustrated in Figure 2.2. To formulate these losses we use three different constraints, namely generative consistency, a symmetry constraint and a discriminator loss. Generative consistency forms the core of the self-supervised constraints to train our viewpoint network and is inspired from the popular analysis-by-synthesis learning paradigm [103]. This framework tries to tackle inverse problems (such as viewpoint estimation) by modelling the forward process of image or feature synthesis. A synthesis function $\mathcal{S}$ models the process of generating an image of an object from a basic representation and a set of parameters. The goal of the analysis function is to infer the underlying parameters which can best explain the formation of an observed input image. Bayesian frameworks such as [191] and inverse graphics [82, 94, 103, 116, 189] form some of the popular techniques that are based on the analysis-by-synthesis paradigm. In our setup, we consider the viewpoint network $\mathcal{V}$ as the analysis function.

We model the synthesis function $\mathcal{S}$, with a viewpoint aware image generation model.

Recent advances in Generative Adversarial Networks (GAN) [29, 93, 131] have shown that it is possible to generate high-quality images with fine-grained control over parameters like appearance, style, viewpoint, etc. Inspired by these works, our synthesis network generates an image, given an input $\boldsymbol{v}$, which controls the viewpoint of the object and an input vector $\boldsymbol{z}$, which controls the style of the object in the synthesized image. By coupling both the analysis ($\mathcal{V}$) and synthesis ($\mathcal{S}$) networks in a cycle, we learn both the networks in a self-supervised manner using cyclic consistency constraints described in 2.3.1 and shown in Figure 2.3. Since the synthesis network can generate high quality images based on controllable inputs $\boldsymbol{v}$ and $\boldsymbol{z}$, these synthesized images can in turn be used as input to the analysis network ($\mathcal{V}$) along with $\boldsymbol{v}$, $\boldsymbol{z}$ as the pseudo ground-truth. On the other hand, for a real world image, if $\mathcal{V}$ predicts the correct viewpoint and style, these can be utilized by $\mathcal{S}$ to produce a similar looking image. This effectively functions as image reconstruction-based supervision. In addition to this, similar to [29, 131] the analysis network also functions as a discriminator, evaluating whether the synthesized images are real or fake. Using a widely prevalent observation that several real-world objects are symmetric, we also enforce a prior constraint via a symmetry loss function to train the viewpoint network. Object symmetry has been used in previous supervised techniques such as [120] for data augmentation, but not as a loss function. In the following, we first describe the various loss constraints used to train the viewpoint network $\mathcal{V}$ while assuming that we already have a trained synthesis network $\mathcal{S}$. In Section 2.4, we describe the loss constraints used to train the synthesis network $\mathcal{S}$.

2.3.1 Generative Consistency

As Figure 2.3 illustrates, we couple the viewpoint network $\mathcal{V}$ with the synthesis network $\mathcal{S}$ to create a circular flow of information resulting in two consistency losses: (a) image consistency and (b) style and viewpoint consistency.

Image consistency Given a real image I sampled from a given image collection $\{\mathbf{I}\}$, we first predict its viewpoint $\hat{\boldsymbol{v}}$ and style code $\hat{\boldsymbol{z}}$ via the viewpoint network $\mathcal{V}$. Then, we pass the predicted $\hat{\boldsymbol{v}}$ and $\hat{\boldsymbol{z}}$ into the synthesis network $\mathcal{S}$ to create the synthetic image $\hat{I}_s$. To train the viewpoint network, we use the image consistency between the input image

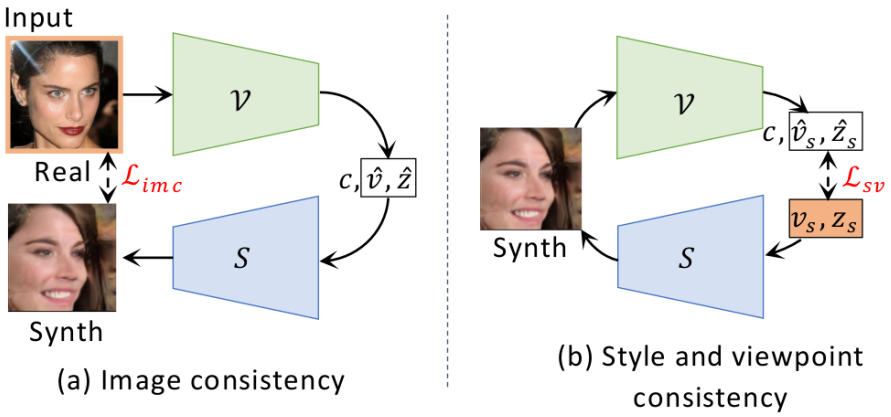


Figure 2.3. Generative consistency. The two cyclic (a) image consistency ($\mathcal{L}_{imc}$) and (b) style and viewpoint consistency ($\mathcal{L}_{sv}$) losses make up generative consistency. The input to each cycle is highlighted in yellow. Image consistency enforces that an input real image, after viewpoint estimation and synthesis, matches its reconstructed synthetic version. Style and viewpoint consistency enforces that the input style and viewpoint provided for synthesis are correctly reproduced by the viewpoint network.

I and corresponding synthetic image I_s with a perceptual loss:

$$\mathcal{L}_{imc} = 1 - \langle \Phi(I), \Phi(\hat{I}_s) \rangle, \tag{2.1}$$

where $\Phi(.)$ denotes the conv5 features of an ImageNet-trained [36] VGG16 classifier [166] and $\langle ., . \rangle$ denotes the cosine distance. Figure 2.3(a) illustrates the image consistency cycle.

Style and viewpoint consistency As illustrated in Figure 2.3(b), we create another circular flow of information with the viewpoint and synthesis networks, but this time starting with a random viewpoint $\boldsymbol{v}_s$ and a style code $\boldsymbol{z}_s$, both sampled from uniform distributions, and input them to the synthesis network to create an image $I_s = \mathcal{S}(\boldsymbol{v}_s, \boldsymbol{z}_s)$. We then pass the synthetic image I_s to the viewpoint network $\mathcal{V}$ that predicts its viewpoint $\hat{\boldsymbol{v}}_s$ and the style code $\hat{\boldsymbol{z}}_s$. We use the sampled viewpoint and style codes for the synthetic image I_s as a pseudo GT to train the viewpoint network. Following [112], the viewpoint consistency loss $\mathcal{L}_v(\hat{\boldsymbol{v}}_1, \hat{\boldsymbol{v}}_2)$ between two viewpoints $\hat{\boldsymbol{v}}_1 = (\hat{a}_1, \hat{e}_1, \hat{t}_1)$ and $\hat{\boldsymbol{v}}_2 = (\hat{a}_2, \hat{e}_2, \hat{t}_2)$ has two components for each Euler angle: (i) cosine proximity between the positive unit vectors $\mathcal{L}_v^{|a|} = -\langle |\hat{a}_1|, |\hat{a}_2| \rangle$ and (ii) the cross-entropy loss $\mathcal{L}_v^{\text{sign}(a)}$ between the classification probabilities of $\text{sign}(\hat{a}_1)$ and $\text{sign}(\hat{a}_2)$. The viewpoint

consistency loss $\mathcal{L}_v$ is a sum of the cross-entropy and cosine proximity losses for all the three Euler angles:

$$\mathcal{L}_v(\hat{\boldsymbol{v}}_1, \hat{\boldsymbol{v}}_2) = \sum_{\phi \in a,e,t} \mathcal{L}_v^{|\phi|} + \mathcal{L}_v^{\text{sign}(\phi)}. \tag{2.2}$$

The overall style and viewpoint loss between the sampled $(\boldsymbol{v}_s, \boldsymbol{z}_s)$ and the predicted $(\hat{\boldsymbol{v}}_s, \hat{\boldsymbol{z}}_s)$ is hence:

$$\mathcal{L}_{sv} = \|\boldsymbol{z}_s - \hat{\boldsymbol{z}}_s\|_2^2 + \mathcal{L}_v(\boldsymbol{v}_s, \hat{\boldsymbol{v}}_s). \tag{2.3}$$

While viewpoint consistency enforces that $\mathcal{V}$ learns correct viewpoints for synthetic images, image consistency helps to ensure that $\mathcal{V}$ generalizes well to real images as well, and hence avoids over-fitting to images synthesized by $\mathcal{S}$.

2.3.2 Discriminator Loss

$\mathcal{V}$ also predicts a score $\hat{\boldsymbol{c}}$ indicating whether an input image is real or synthetic. It thus acts as a discriminator in a typical GAN [55] setting, helping the synthesis network create more realistic images. We use the discriminator loss from Wasserstein-GAN [7] to update the viewpoint network using:

$$\mathcal{L}_{dis} = -\mathbb{E}_{x \sim p_{\text{real}}}[\boldsymbol{c}] + \mathbb{E}_{\hat{x} \sim p_{\text{synth}}}[\hat{\boldsymbol{c}}], \tag{2.4}$$

where $\boldsymbol{c} = \mathcal{V}(x)$ and $\hat{\boldsymbol{c}} = \mathcal{V}(\hat{x})$ are the predicted class scores for the real and the synthesized images, respectively.

2.3.3 Symmetry Constraint

Symmetry is a strong prior observed in many commonplace object categories, *e.g.*, faces, boats, cars, airplanes, etc. For categories with symmetry, we propose to leverage an additional symmetry constraint. Given an image I of an object with viewpoint (a, e, t), the GT viewpoint of the object in a horizontally flipped image $flip(I)$ is given by (-a, e,-t). We enforce a symmetry constraint on the viewpoint network's outputs $(\hat{\boldsymbol{v}}, \hat{\boldsymbol{z}})$ and $(\hat{\boldsymbol{v}}^*, \hat{\boldsymbol{z}}^*)$ for a given image I and its horizontally flipped version $flip(I)$, respectively. Let $\hat{\boldsymbol{v}}$=$(\hat{a}, \hat{e}, \hat{t})$ and $\hat{\boldsymbol{v}}^*$=$(\hat{a}^*, \hat{e}^*, \hat{t}^*)$ and we denote the flipped viewpoint of the flipped image as $\hat{\boldsymbol{v}}_f^*$=(-$\hat{a}^*$, $\hat{e}^*$,-$\hat{t}^*$). The symmetry loss is given as

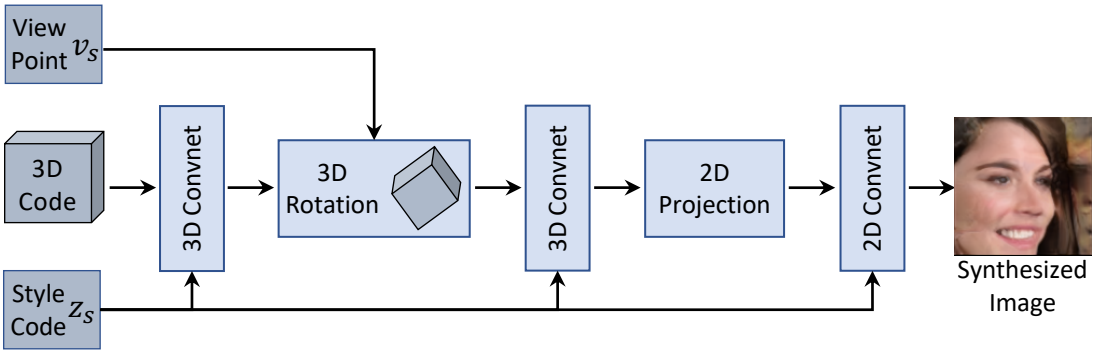


Figure 2.4. Synthesis network overview. The network takes viewpoint $\boldsymbol{v}_s$ and style code $\boldsymbol{z}_s$ to produce a viewpoint aware image.

$$\mathcal{L}_{sym} = \mathcal{D}(\hat{\boldsymbol{v}}, \hat{\boldsymbol{v}}_f^*) + \|\hat{\boldsymbol{z}} - \hat{\boldsymbol{z}}^*\|_2^2 . \tag{2.5}$$

Effectively, for a given horizontally flipped image pair, we regularize that the network predicts similar magnitudes for all the angles and opposite directions for azimuth and tilt. Additionally, the above loss enforces that the style of the flipped image pair is consistent.

Our overall loss to train the viewpoint network $\mathcal{V}$ is a linear combination of the aforementioned loss functions:

$$\mathcal{L}_V = \lambda_1 \mathcal{L}_{sym} + \lambda_2 \mathcal{L}_{imc} + \lambda_3 \mathcal{L}_{sv} + \lambda_4 \mathcal{L}_{dis}, \tag{2.6}$$

where the parameters $\{\lambda_i\}$ determine the relative importance of the different losses, which we empirically determine using a grid search.

2.4 Viewpoint-Aware Synthesis Network

Recent advances in GANs such as InfoGAN [29], StyleGAN [93] and HoloGAN [131] demonstrate the possibility of conditional image synthesis where we can control the synthesized object's attributes such as object class, viewpoint, style, geometry, etc. A key insight that we make use of in our synthesis network and which is also used in recent GANs such as HoloGAN [131] and other works[101, 167, 201], is that one can instill 3D geometric meaning into the network's latent representations by performing explicit geometric transformations such as rotation on them. A similar idea has also been used successfully with other generative models such as auto-encoders [69, 142, 153]. Our

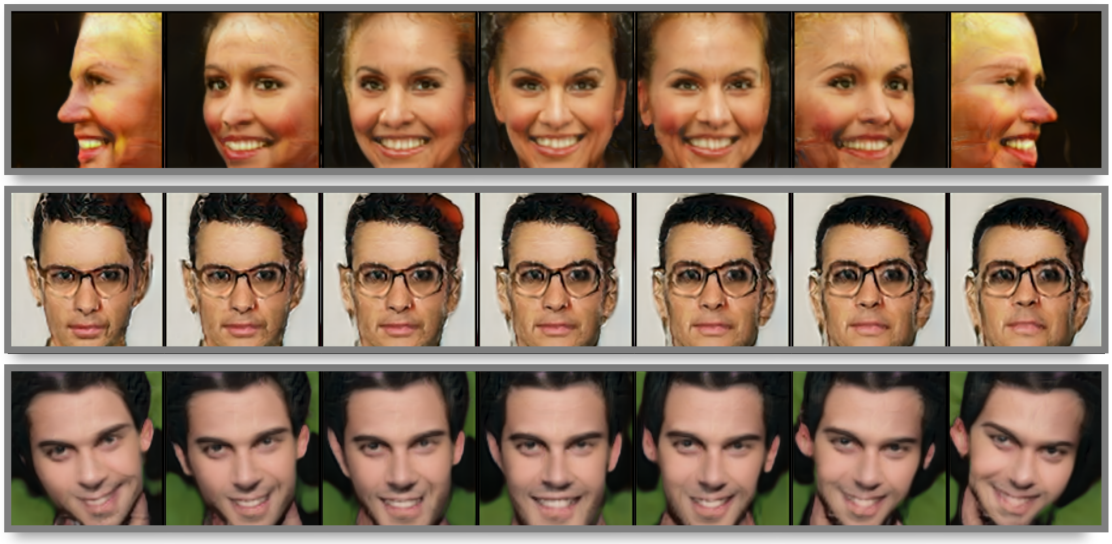
viewpoint-aware synthesis network has a similar architecture to HoloGAN [131], but is tailored for the needs of viewpoint estimation. HoloGAN is a pure generative model with GAN losses to ensure realism and an identity loss to reproduce the input style code, but lacks a corresponding viewpoint prediction network. In this work, since we focus on viewpoint estimation, we introduce tight coupling of HoloGAN with a viewpoint prediction network and several novel loss functions to train it in a manner that is conducive to accurate viewpoint prediction.

Synthesis network overview Figure 3.3 illustrates the design of the synthesis network. The network $\mathcal{S}$ takes a style code $\boldsymbol{z}_s$ and a viewpoint $\boldsymbol{v}_s$ to produce a corresponding object image I_s. The goal of $\mathcal{S}$ is to learn a disentangled 3D representation of an object, which can be used to synthesize objects in various viewpoints and styles, hence aiding in the supervision of the viewpoint network $\mathcal{V}$. We first pass a learnable canonical 3D latent code through a 3D network, which applies 3D convolutions to it. Then, we rotate the resulting 3D representation with $\boldsymbol{v}_s$ and pass through an additional 3D network. We project this viewpoint-aware learned 3D code on to 2D using a simple orthographic projection unit. Finally, we pass the resulting 2D representation through a StyleGAN [93]-like 2D network to produce a synthesized image. The style and appearance of the image is controlled by the sampled style code $\boldsymbol{z}_s$. Following StyleGAN [93], the style code $\boldsymbol{z}_s$ affects the style of the resulting image via adaptive instance normalization [72] in both the 3D and 2D representations. For stable training, we freeze $\mathcal{V}$ while training $\mathcal{S}$ and vice versa.

Loss functions Like the viewpoint network, we use several constraints to train the synthesis network, which are designed to improve viewpoint estimation. The first is the standard adversarial loss used in training Wasserstein-GAN[7]:

$$\mathcal{L}_{adv} = -\mathbb{E}_{\hat{x}\sim p_{\text{synth}}}[\hat{\boldsymbol{c}}] \tag{2.7}$$

where $\hat{\boldsymbol{c}} = \mathcal{V}(\hat{x})$ is the class membership score predicted by $\mathcal{V}$ for a synthesized image. The second is a paired version of the style and viewpoint consistency loss (Eqn. 2.3) described in Section 2.3.1, where we propose to use multiple paired $(\boldsymbol{z}_s, \boldsymbol{v}_s)$ samples to enforce style and viewpoint consistency and to better disentangle the latent representa-



(a) Faces

(b) Cars

Figure 2.5. Synthesis results. Example synthetic images of (a) faces and (b) cars generated by the viewpoint-aware generator $\mathcal{S}$. For each row the style vector z is constant, whereas the viewpoint is varied monotonically along the azimuth (first row), elevation (second row) and tilt (third row) dimensions.

tions of $\mathcal{S}$. The third is a flip image consistency loss. Note that, in contrast to our work, InfoGAN [29] and HoloGAN [131] only use adversarial and style consistency losses.

Style and viewpoint consistency with paired samples Since we train the viewpoint network with images synthesized by $\mathcal{S}$, it is very important for $\mathcal{S}$ to be sensitive and responsive to its input style $\boldsymbol{z}_s$ and viewpoint $\boldsymbol{v}_s$ parameters. An ideal $\mathcal{S}$ would perfectly disentangle $\boldsymbol{v}_s$ and $\boldsymbol{z}_s$. That means, if we fix $\boldsymbol{z}_s$ and vary $\boldsymbol{v}_s$, the resulting object images

should have the same style, but varying viewpoints. On the other hand, if we fix $\boldsymbol{v}_s$ and vary $\boldsymbol{z}_s$, the resulting object images should have different styles, but a fixed viewpoint. We enforce this constraint with a paired version of the style and viewpoint consistency (Eqn. 2.3) loss where we sample 3 different pairs of $(\boldsymbol{z}_s, \boldsymbol{v}_s)$ values by varying one parameter at a time as: $\{(\boldsymbol{z}_s^0, \boldsymbol{v}_s^0), (\boldsymbol{z}_s^0, \boldsymbol{v}_s^1), (\boldsymbol{z}_s^1, \boldsymbol{v}_s^1)\}$. We refer to this paired style and viewpoint loss as $\mathcal{L}_{sv,pair}$. The ablation study in Section 2.5 suggests that this paired style and viewpoint loss helps to train a better synthesis network for our intended task of viewpoint estimation. We also observe qualitatively that the synthesis network successfully disentangles the viewpoints and styles of the generated images. Some example images synthesized by $\mathcal{S}$ for faces and cars are shown in Figure 2.5. Each row uses a fixed style code $\boldsymbol{z}_s$ and we monotonically vary the input viewpoint $\boldsymbol{v}_s$ by changing one of its a, e or t values across the columns.

Flip image consistency This is similar to the symmetry constraint used to train the viewpoint network, but applied to synthesized images. Flip image consistency forces $\mathcal{S}$ to synthesize horizontally flipped images when we input appropriately flipped viewpoints. For the pairs $\mathcal{S}(\boldsymbol{v}_s, \boldsymbol{z}_s) = I_s$ and $\mathcal{S}(\boldsymbol{v}_s^*, \boldsymbol{z}_s) = I_s^*$, where $\boldsymbol{v}^*$ has opposite signs for the a and t values of $\boldsymbol{v}_s$, the flip consistency loss is defined as:

$$\mathcal{L}_{fc} = \|I_s - \text{flip}(I_s^*)\|_1 \tag{2.8}$$

where $\text{flip}(I_s^*)$ is the horizontally flipped version of I_s^*.

The overall loss for the synthesis network is given by:

$$\mathcal{L}_S = \lambda_5 \mathcal{L}_{adv} + \lambda_6 \mathcal{L}_{sv,pair} + \lambda_7 \mathcal{L}_{fc} \tag{2.9}$$

where the parameters $\{\lambda_i\}$ are the relative weights of the losses which we determine empirically using grid search.

2.5 Experiments

We empirically validate our approach with extensive experiments on head pose estimation and viewpoint estimation on other object categories of buses, cars and trains. We

refer to our approach as 'SSV'.

Implementation and training details We implement our framework in Pytorch[143]. We provide all network architecture details, and run-time and memory analyses in the supplementary material.

Viewpoint calibration The output of SSV for a given image I is $(\hat{a}, \hat{e}, \hat{t})$. However, since SSV is self-supervised, the co-ordinate system for predictions need not correspond to the actual canonical co-ordinate system of GT annotations. For quantitative evaluation, following the standard practice in self-supervised learning of features [22, 38, 196] and landmarks [75, 174, 198], we fit a linear regressor that maps the predictions of SSV to GT viewpoints using 100 randomly chosen images from the target test dataset. Note that this calibration with a linear regressor only rotates the predicted viewpoints to the GT canonical frame of reference. We do not update or learn our SSV network during this step.

2.5.1 Head Pose Estimation

Human faces have a special place among objects for viewpoint estimation and head pose estimation has attracted considerable research attention [20, 25, 60, 102, 170, 186, 187, 202]. The availability of large-scale datasets [41, 159] and the existence of ample research provides a unique opportunity to perform extensive experimental analysis of our technique on head pose estimation.

Datasets and evaluation metric For training, we use the 300W-LP [159] dataset, which combines several in-the-wild face datasets. It contains 122,450 face images with diverse viewpoints, created by fitting a 3D face morphable model [16] to face images and rendering them from various viewpoints. Note that we only use the images from this dataset to train SSV and not their GT viewpoint annotations. We evaluate our framework on the BIWI [41] dataset which contains 15,677 images across 24 sets of video sequences of 20 subjects in a wide variety of viewpoints. We use the MTCNN face detector to detect all faces [195]. We compute average absolute errors (AE) for azimuth, elevation and tilt between the predictions and GT. We also report the mean absolute error (MAE) of these

	Method	Azimuth	Elevation	Tilt	MAE
Self-Supervised	LMDIS [198] + PnP	16.8	26.1	5.6	16.1
	IMM [79] + PnP	14.8	22.4	5.5	14.2
	SCOPS [75] + PnP	15.7	13.8	7.3	12.3
	HoloGAN [131]	8.9	15.5	5.0	9.8
	HoloGAN [131] with v	7.0	15.1	5.1	9.0
	SSV w/o $\mathcal{L}_{sym}$ + $\mathcal{L}_{imc}$	6.8	13.0	5.2	8.3
	SSV w/o $\mathcal{L}_{imc}$	6.9	10.3	**4.4**	7.2
	SSV-Full	**6.0**	**9.8**	**4.4**	**6.7**
Supervised	3DDFA [202]	36.2	12.3	8.7	19.1
	KEPLER [102]	8.8	17.3	16.2	13.9
	DLib [96]	16.8	13.8	6.1	12.2
	FAN [20]	8.5	7.4	7.6	7.8
	Hopenet [158]	5.1	6.9	3.3	5.1
	FSA [186]	**4.2**	**4.9**	**2.7**	**4.0**

Table 2.1. Head pose estimation ablation studies and SOTA comparisons. Average absolute angular error for azimuth, elevation and tilt Euler angles in degrees together with the mean absolute error (MAE) for the BIWI [41] dataset.

three errors.

Ablation study We empirically evaluate the different self-supervised constraints used to train the viewpoint network. Table 2.1 shows that for head pose estimation, using all the proposed constraints (SSV-Full) results in our best MAE of $6.7°$. Removing the image consistency constraint $\mathcal{L}_{imc}$ leads to an MAE to $7.2°$ and further removing the symmetry constraint $\mathcal{L}_{sym}$ results in an MAE of $8.3°$. These results demonstrate the usefulness of the generative image consistency and symmetry constraints in our framework.

Additionally, we evaluate the effect of using the paired style and viewpoint loss $\mathcal{L}_{sv,pair}$ to train the viewpoint-aware synthesis network $\mathcal{S}$. We observe that when we train $\mathcal{S}$ without $\mathcal{L}_{sv,pair}$, our viewpoint network (SSV-full model) results in AE values of $7.8°$ (azimuth), $11.1°$ (elevation), $4.2°$ (tilt) and an MAE of $7.7°$. This represents a $1°$ increase from the corresponding MAE value of $6.7°$ for the SSV-full, where $\mathcal{S}$ is trained with $\mathcal{L}_{sv,pair}$ (Table 2.1, SSV-full). This shows that our paired style and viewpoint loss helps to better train the image synthesis network for the task of viewpoint estimation.

	Method	Azimuth	Elevation	Tilt	MAE
Self-Sup	SSV non-refined	6.9	9.4	**4.2**	6.8
	SSV refined on BIWI	**4.9**	**8.5**	**4.2**	**5.8**
Supervised	FSA [186]	**2.8**	4.2	3.6	**3.6**
	DeepHP [126]	5.6	5.1	-	-
	RNNFace [60]	3.9	**4.0**	**3.0**	**3.6**

Table 2.2. Improved head pose estimation with fine-tuning. Average angular error for each of the Euler angles together with mean average error (MAE) on data of 30% held-out sequences of the BIWI [41] dataset and fine-tuning on the remaining 70% without using their annotations. All values are in degrees.

Comparison with self-supervised methods Since SSV is a self-supervised viewpoint estimation work, there is no existing work that we can directly compare against. One could also obtain head pose from predicted face landmarks and we compare against recent state-of-the-art self-supervised landmark estimation (LMDIS [198], IMM [79]) and part discovery techniques (SCOPS [75]). We fit a linear regressor that maps the self-supervisedly learned semantic face part centers from SCOPS and landmarks from LMDIS, IMM to five canonical facial landmarks (left-eye center, right-eye center, nose tip and mouth corners). Then we fit an average 3D face model to these facial landmarks with the Perspective-n-Point (PnP) algorithm [106] to estimate head pose. We also quantify HoloGAN's [131] performance at viewpoint estimation, by training a viewpoint network with images synthesized by it under different input viewpoints (as pseudo GT). Alternatively, we train HoloGAN with an additional viewpoint output and a corresponding additional loss for it. For both these latter approaches, we additionally use viewpoint calibration, similar to SSV. We consider these works as our closest baselines because of their self-supervised training. The MAE results in Table 2.1 indicate that SSV performs considerably better than all the competing self-supervised methods.

Comparison with supervised methods As a reference, we also report the metrics for the recent state-of-the-art fully-supervised methods. Table 2.1 shows the results for both the keypoint-based [20, 96, 102, 202] and keypoint-free [158, 186] methods. The latter methods learn to directly regress head orientation values from networks. The results indicate that 'SSV-Full', despite being purely self-supervised, can obtain comparable

results to fully supervised techniques. In addition, we notice that SSV-Full (with MAE $6.7°$) outperforms all the keypoint-based supervised methods [20, 96, 102, 202], where FAN [20] has the best MAE of $7.8°$.

Refinement on BIWI dataset The results reported thus far are with training on the 300W-LP [159] dataset. Following some recent works [60, 126, 186], we use 70% (16) of the image sequences in the BIWI dataset to fine-tune our model. Since our method is self-supervised, we just use images from BIWI without the annotations. We use the remaining 30% (8) image sequences for evaluation. The results of our model along with those of the state-of-the-art supervised models are reported in Table 2.2. After refinement with the BIWI dataset's images, the MAE of SSV significantly reduces to $5.8°$. This demonstrates that SSV can improve its performance with the availability of images that match the target domain, even without GT annotations. We also show qualitative results of head pose estimation for this refined SSV-Full model in Figure 2.6(a). It performs robustly to large variations in head pose, identity and expression.

2.5.2 Generalization to Other Object Categories

SSV is not specific to faces and can be used to learn viewpoints of other object categories. To demonstrate its generalization ability, we additionally train and evaluate SSV on the categories of cars, buses and trains.

Datasets and evaluation metric Since SSV is completely self-supervised, the training image collection has to be reasonably large to cover all possible object viewpoints while covering diversity in other image aspects such as appearance, lighting etc. For this reason, we leverage large-scale image collections from both the existing datasets and the internet to train our network. For the car category, we use the CompCars [185] dataset, which is a fine-grained car model classification dataset containing 137,000 car images in various viewpoints. For the 'train' and 'bus' categories, we use the OpenImages [11, 137, 138] dataset which contains about 12,000 images of each of these categories. Additionally, we mine about 30,000 images from Google image search for each category. None of the aforementioned datasets have viewpoint annotations. This also

demonstrates the ability of SSV to consume large-scale internet image collections that come without any viewpoint annotations.

We evaluate the performance of the trained SSV model on the test sets of the challenging Pascal3D+ [183] dataset. The images in this dataset have extreme shape, appearance and viewpoint variations. Following [112, 120, 147, 177], we estimate the azimuth, elevation and tilt values, given the GT object location. To compute the error between the predicted and GT viewpoints, we follow the standard geodesic distance $\Delta(R_{gt}, R_p) = \left\|\log R_{gt}^T R_p\right\|_{\mathcal{F}} / \sqrt{2}$ between the predicted rotation matrix R_p constructed using viewpoint predictions and R_{gt} constructed using GT viewpoints [112]. Using this distance metric, we report the median geodesic error (Med. Error) for the test set. Additionally, we also compute the percentage of inlier predictions whose error is less than $\pi/6$ (Acc@$\pi/6$).

(a) Faces (b) Cars (c) Buses (d) Trains

Figure 2.6. Viewpoint estimation results. We visually show the results of (a) head pose estimation on the BIWI [41] dataset and of viewpoint estimation on the test sets of the (b) car, (c) bus and (d) train categories from the PASCAL3D+ [183] dataset. Solid arrows indicate predicted viewpoints, while the dashed arrows indicate their GT values. Our self-supervised method performs well for a wide range of head poses, identities and facial expressions. It also successfully handles different object appearances and lighting conditions from the car, bus and train categories. We show additional results in the supplementary material.

Baselines For head pose estimation, we compared with self-supervised landmark [75, 79, 198] discovery techniques coupled with the PnP algorithm for head pose estimation by fitting them to an average 3D face. For objects like cars with full 360° azimuth rotation, we notice that the landmarks produced by SCOPS [75] and LMDIS [198] cannot be used for reasonable viewpoint estimates. This is because SCOPS is primarily a self supervised part segmentation framework which does not distinguish between front and rear parts of the car. Since the keypoints we compute are the centers of part segments, the

resulting keypoints cannot distinguish such parts. LMDIS on the other hand produces keypoints only for the side profiles of cars. Hence, we use another baseline technique for comparisons on cars, trains and buses. Following the insights from [75, 174] that features learned by image classification networks are equivariant to object rotation, we learn a linear regressor that maps the Conv5 features of a pre-trained VGG network [166] to the viewpoint of an object. To train this baseline, we use the VGG image features and the GT viewpoint annotations in the Pascal3D+ training dataset [183]. We use the same Pascal3D+ annotations used to calibrate SSV's predicted viewpoints to GT canonical viewpoint axes. We consider this as a self-supervised baseline since we are not using GT annotations for feature learning but only to map the features to viewpoint predictions. We refer to this baseline as VGG-View. As an additional baseline, we train HoloGAN [131] with an additional viewpoint output and a corresponding loss for it. The viewpoint predictions are calibrated, similar to SSV.

Comparisons We compare SSV to our baselines and also to several state-of-the-art supervised viewpoint estimation methods on the Pascal3D+ test dataset. Table 2.3 indicates that SSV significantly outperforms the baselines. With respect to supervised methods, SSV performs comparably to Tulsiani *et al*. [177] and Mahendran *et al*. [120] in terms of Median error. Interestingly for the 'train' category, SSV performs even better than supervised methods. These results demonstrate the general applicability of SSV for viewpoint learning on different object categories. We show some qualitative results for these categories in Figure 2.6(b)-(d).

2.6 Conclusions

In this work we investigate the largely unexplored problem of learning viewpoint estimation in a self-supervised manner from collections of un-annotated object images. We design a viewpoint learning framework that receives supervision from a viewpoint-aware synthesis network; and from additional symmetry and adversarial constraints. We further supervise our synthesis network with additional losses to better control its image synthesis process. We show that our technique outperforms existing self-supervised techniques and performs competitively to fully-supervised ones on several object categories like faces, cars, buses and trains.

	Method	Car	Bus	Train
Self-Sup	VGG-View	34.2	19.0	9.4
	HoloGAN [131] with v	16.3	14.2	9.7
	SSV-Full	**10.1**	**9.0**	**5.3**
Supervised	Tulsiani *et al*. [177]	9.1	5.8	8.7
	Mahendran *et al*. [120]	8.1	4.3	7.3
	Liao *et al*. [112]	5.2	3.4	**6.1**
	Grabner *et al*. [58]	**5.1**	**3.3**	6.7

Table 2.3. Generalization to other object categories, median error. We show the median geodesic errors (in degrees) for the car, bus and train categories.

	Method	Car	Bus	Train
Self-sup	VGG-View	0.43	0.69	0.82
	HoloGAN [131] with v	0.52	0.73	0.81
	SSV-Full	**0.67**	**0.82**	**0.96**
Supervised	Tulsiani *et al*. [177]	0.89	**0.98**	0.80
	Mahendran *et al*. [120]	-	-	-
	Liao *et al*. [112]	**0.93**	0.97	**0.84**
	Grabner *et al*. [58]	**0.93**	0.97	0.80

Table 2.4. Generalization to other object categories, inlier count. We show the percentage of images with geodesic error less than $\pi/6$ for the car, bus and train categories.

Chapter 3

Self-Supervised Object Detection via Generative Image Synthesis

3.1 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 [113, 150], 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 [151, 152, 192], 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 [203]. The second paradigm is of rendering-based methods, including Meta-Sim [90] and Meta-Sim2 [37], which learn object detection from synthetically rendered images. Creating them, however, requires large collections of high-quality 3D CAD

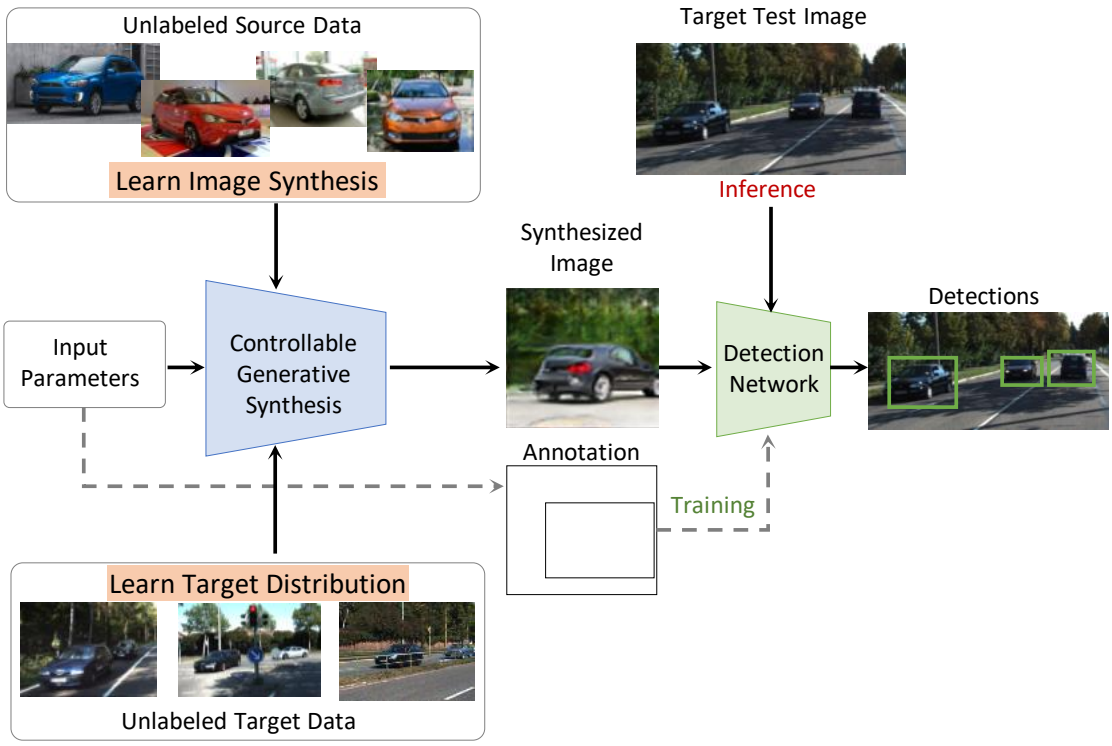


Figure 3.1. Self-Supervised Object Detection. We learn object detection purely using natural image collections without bounding box labels. We leverage controllable GANs to synthesize images and to detect objects together in a tightly coupled framework. We learn image synthesis from unlabeled singe-object source images (*e.g.*, Compcars [185]) and optimally adapt our framework to any multi-object unlabeled target dataset (*e.g.*, KITTI [48]).

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.

Recently, there has been much progress in making Generative Adversarial Networks (GANs) [56] controllable using input parameters like shape, viewpoint, position and keypoints [131, 132, 134, 161], 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 [127] and keypoints [79, 174, 198] 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. 3.1). 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 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 [48] and Cityscapes [33] datasets for car object detection. SSOD outperforms the best prior image-based self-supervised object detection method Wetectron [152] 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 [37]. 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 $\sim$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 [37].

3.2 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 [79, 174, 198] and part segmentation [32, 75]. [12] 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 [87, 100, 108, 109], viewpoint estimation [127] and point cloud estimation [128]. 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 [40, 131, 132, 134, 161] or implicit representations [122, 156, 194]. Some of these works have been used in analysis-by-synthesis frameworks to solve computer vision tasks, including 3D reconstruction [65, 66, 108, 109], viewpoint estimation [127] and keypoint estimation [79]. 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 [13, 46, 151, 172, 192], discriminative [165], curriculum [89, 152, 197] or self-taught [85] learning framework. However, such methods rely heavily on object proposals generated by methods like [6, 178, 203], 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 [21, 37, 45, 90, 146, 154] learn object detection through synthetic data from graphics renderers. [154] obtains synthetic images and annotations from a game engine. In [21, 45], the authors exactly mimic real world dataset (*e.g.*, KITTI [48]) in a synthetic simulator. In [146], the authors synthesize scenes by randomizing location, orientations and textures of objects of interest in a scene. In Meta-Sim [90] and Meta-Sim2 [37], the authors propose a strategy to learn optimal scene parameters to generate images similar to a target dataset. While methods like [21, 45] use annotations from real world datasets to mimic the datasets in synthetic worlds, other methods like [37, 146, 154] 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.

3.3 Self-Supervised Object Detection

3.3.1 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. 3.1). 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

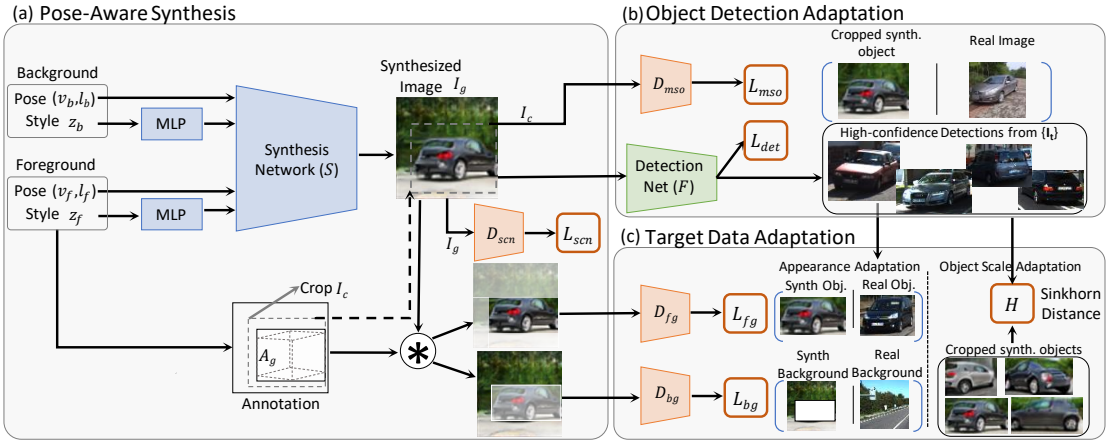


Figure 3.2. Overview of Self-Supervised Object Detection. SSOD contains three modules: (a) a pose-aware synthesis module that generates images with objects in predefined poses using a controllable GAN for training object detectors; (b) an object detection adaptation module that guides the synthesis process 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.

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. 3.1). We discuss more about the need for this assumption in Sec 3.3.3. 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.

3.3.2 Overview of SSOD

We present an overview of SSOD in Fig. 3.2. 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. 3.2(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. 3.2(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. 3.2(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.

3.3.3 Pose-Aware Synthesis

Our pose-aware synthesis network $\mathcal{S}$ is inspired by the recent BlockGAN [132], 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. 3.3. To make BlockGAN [132] 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. 3.3).

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 initial-

ized 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. 3.3). 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 [132] generates images at a resolution of 64×64. For our $\mathcal{S}$, we modify it and adopt the strategy of progressive growing of GANs [91, 92] to increase its synthesis resolution to 256×256.

We train $\mathcal{S}$ in a GAN setup with an adversarial loss [7] $\mathcal{L}_{scn}$ computed using a scene discriminator $\mathcal{D}_{scn}$ as:

$$\mathcal{L}_{scn} = -\mathbb{E}_{I_g \sim p_{\text{synth}}}[\mathcal{D}_{scn}(I_g)], \tag{3.1}$$

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 [132] 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).

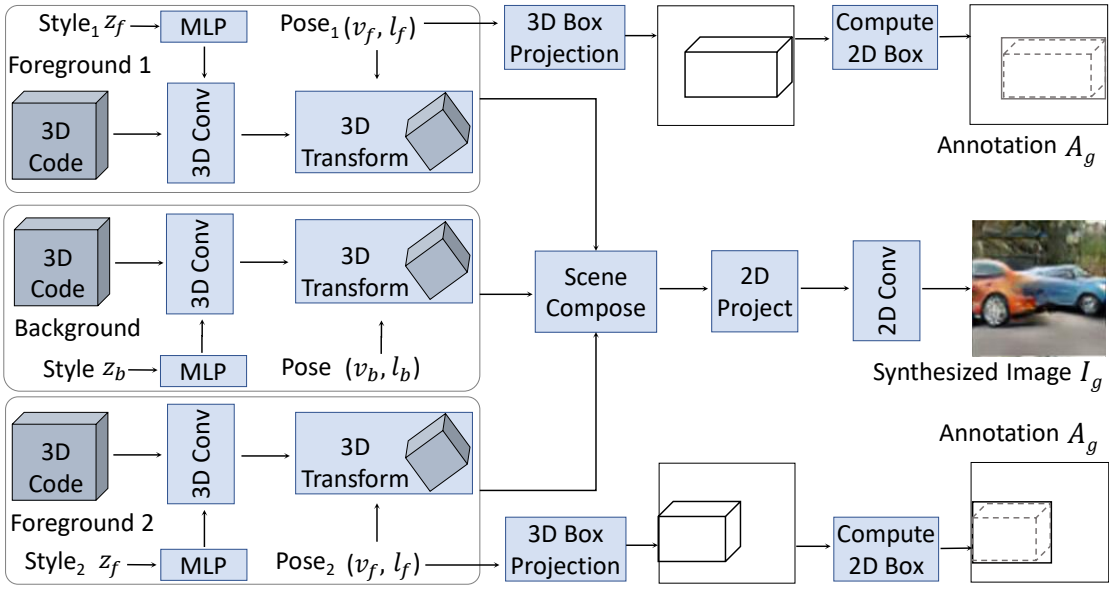


Figure 3.3. Pose-Aware Synthesis Network ($\mathcal{S}$) Overview. $\mathcal{S}$ takes as input separate style codes ($\boldsymbol{z}$) and poses ($\boldsymbol{v}$, $\boldsymbol{l}$) for the background and one/more foreground objects; transforms their respective learned 3D codes with the provided poses; 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.

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. 3.3. The paired data $\langle I_g, A_g \rangle$ can then be used to train the object detection network $\mathcal{F}$.

3.3.4 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.

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. 3.2(b)). The object detection network $\mathcal{F}$ is a standard Feature Pyramid Network [113], which takes 2D images as input and predicts bounding boxes for the object. It is trained using the standard object detection losses ($\mathcal{L}_{det}$) [113]. 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 $\mathcal{S}$.

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. 3.2(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:

$$\mathcal{L}_{mso} = -\mathbb{E}_{I_c \sim p_{\text{synth}}}[\mathcal{D}_{mso}(I_c)], \tag{3.2}$$

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

3.3.5 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. 3.1 and Fig. 3.4). 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. 3.2(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.

Foreground Appearance Loss

We compute the foreground appearance loss $\mathcal{L}_{fg}$ via a patch-based [78] discriminator $\mathcal{D}_{fg}$ (Fig. 3.2(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:

$$\mathcal{L}_{fg} = -\mathbb{E}_{I_g \sim p_{\text{synth}}}[\hat{c}_{fg}] * M_g, \tag{3.3}$$

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. 3.3.2). 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:

$$\mathcal{L}_{d_{fg}} = -\mathbb{E}_{I_t \sim p_{\text{real}}}[c_t] * M_t + \mathbb{E}_{I_g \sim p_{\text{synth}}}[\hat{c}_{fg}] * M_g, \tag{3.4}$$

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

Background Appearance Loss

The background discriminator $\mathcal{D}_{bg}$ is also a patch-based discriminator (Fig. 3.2(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

$$\mathcal{L}_{bg} = -\mathbb{E}_{I_g \sim p_{\text{synth}}}[\hat{c}_{bg}] * (1 - M_g), \tag{3.5}$$

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 [162]. 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:

$$\mathcal{L}_{d_{bg}} = -\mathbb{E}_{I_t^b \sim p_{\text{real}}}[c_t^b] + \mathbb{E}_{I_g \sim p_{\text{synth}}}[\hat{c}_{bg}] * (1 - M_g), \tag{3.6}$$

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. 3.3), 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. 3.3.3). Empirically, we observe this is effective enough in learning the foreground and background distributions of the target domain.

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:

$$d_o = \underset{d_i}{\operatorname{argmin}} \mathcal{H}(\Phi(\alpha^{d_i}), \Phi(\beta^{d_i})), \tag{3.7}$$

where Φ computes the conv5 features of a pre-trained image classification VGG [166]) network and $\mathcal{H}$ is the Sinkhorn distance [35] 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.

3.3.6 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 [162] (described in Sec. 3.3.5) 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:

$$\begin{aligned}\mathcal{L}_{syn} =& \lambda_{scn}\,\mathcal{L}_{scn} + \lambda_{mso}\,\mathcal{L}_{mso} + \lambda_{det}\,\mathcal{L}_{det} \\ &+ \lambda_{fg}\,\mathcal{L}_{fg} + \lambda_{bg}\,\mathcal{L}_{bg},\end{aligned} \tag{3.8}$$

where $\{\lambda_i\}$ are the relative weights of the various losses. Lastly, as discussed in Sec. 3.3.5 we find the optimal set of input object depth parameters for $\mathcal{S}$ that align synthesized data further to the target distribution.

3.4 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.

3.4.1 Datasets and Evaluation

We use three datasets containing images of car objects to train and evaluate SSOD: (a) the Compcars dataset [185] as the single-car source dataset and (b) two multi-car KITTI [48] and Cityscapes [33] target datasets containing outdoor driving scenes. During training, we do not use bounding box annotations for any of these datasets.

Compcars. The Compcars dataset [185] 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. 3.1). 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 [48] 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. 3.1). 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 [48] of the its *validation* set.

Method	Coupled	Easy ↑	Medium ↑	Hard ↑	All ↑	Sinkhorn [35] ↓	KID [15] ↓	FID [67] ↓
BlockGAN [132] 64	✗	65.1	48.3	40.5	51.3	0.486	0.048	8.3
BlockGAN [132] 128	✗	69.4	49.9	44.2	54.5	0.483	0.046	7.8
BlockGAN [132] 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 3.1. 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.

Cityscapes. Similarly to KITTI, we also evaluate SSOD on the challenging Cityscapes [33] outdoor driving target dataset with images of size 512×1024. We use the version provided by [44] containing bounding box annotations. We split it into disjoint *training* (3000 unlabeled images) and *validation* (1000 labeled images) sets as provided in [44].

3.4.2 Ablation Study

We conduct ablation studies on the KITTI dataset to evaluate the contribution of each individual component of SSOD (Table 3.1). We evaluate object detection performance using mAP, and compute SinkHorn [35], KID [15] and FID [67] 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 [132] 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. 3.3.3. The results are shown in the top three rows of Table 3.1. 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. 3.4. 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 ($\mathcal{S}$) and object detection ($\mathcal{F}$) networks. We evaluate four variants of SSOD: (a) without the target data appearance adaption losses described in Sec. 3.3.5 (SSOD *w/o* $\mathcal{L}_{fg}$ + $\mathcal{L}_{bg}$); (b) without the multi-scale object synthesis loss $\mathcal{L}_{mso}$ described in Sec. 3.3.4 (SSOD *w/o* $\mathcal{L}_{mso}$); (c) without adaptation to the target dataset's object scales as described in Sec. 3.3.5 (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 3.1) perform significantly better than those without (top three rows of Table 3.1). 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. 3.4 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. 3.4(a) shows the images

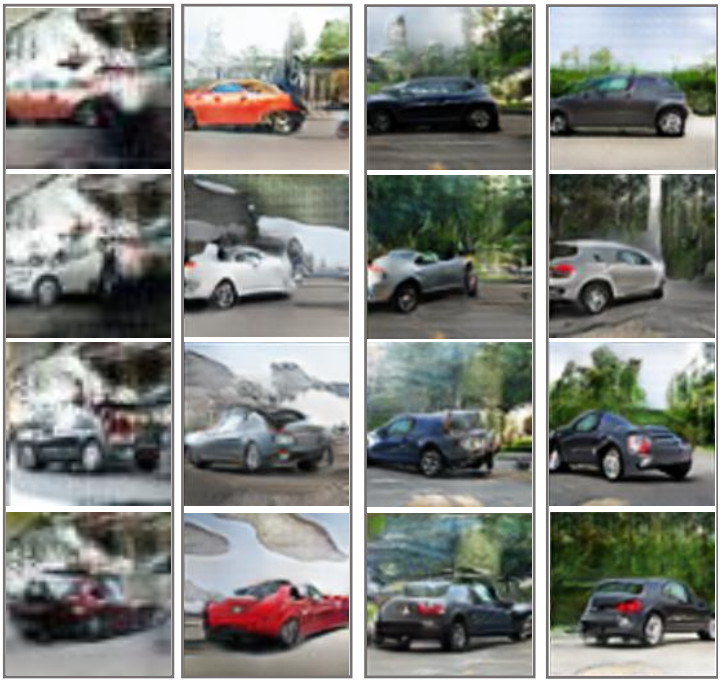


Figure 3.4. Qualitative analysis of image synthesis. The columns show images generated by (a) BlockGAN [132] at 64×64; and by $\mathcal{S}$ for (b) SSOD trained without $\mathcal{L}_{fg}$, $\mathcal{L}_{bg}$, and $\mathcal{L}_{mso}$; (c) SSOD trained without $\mathcal{L}_{mso}$; and (d) the full SSOD model. Each row has images generated with the same pose, and foreground and background style codes. Rows (b)-(d) show 256×256 sized images.

synthesized by the original BlockGAN [132] at a resolution of 64×64 suffers from poor quality. Fig. 3.4(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. 3.4(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 3.2.

Method	3D Assets	Easy↑	Medium↑	Hard↑	All↑
PCL [172]	✗	47.3	32.9	19.4	33.2
Wetectron [152]	✗	51.3	37.9	25.1	38.1
SSOD-Full (ours)	✗	**80.8**	**68.1**	**56.6**	**68.4**
Meta-Sim* [90]	✓	65.9	66.3	66.0	66.0
Meta-Sim2 [37]	✓	67.0	67.0	66.2	66.7

Table 3.2. Comparisons to SOTA. Object detection performance (mAP at IOU 0.5) on KITTI of SSOD and various SOTA methods.

3.4.3 Comparisons to State-of-the-Art

On the KITTI dataset, we compare SSOD to existing methods, Wetectron [152] and PCL [172], 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 [152] is the best-performing prior method. We train Wetectron and PCL with a combination of Compcars [185] and KITTI's [48] *training* set; use image-level labels for the presence/absence of the object; get object proposals from Edgeboxes [203]; and evaluate it on KITTI's *validation* set. The results are in Table 3.2. 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 [90] and Meta-Sim2 [37]. 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-

*We report detection accuracy values for the version of Meta-Sim that does not use labeled validation images from the KITTI [48] dataset.

Method	mAP↑	Sinkhorn↓
Wetectron [152]	18.2	0.549
BlockGAN [132] 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 3.3. Performance on Cityscapes. Object detection performance (mAP at IOU 0.5) and synthetic data quality analysis (Sinkorn) on Cityscapes.

Sim2 for Easy, Medium and All cases in KITTI (Table 3.2). 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. 3.4.5). 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.

3.4.4 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 [33]. We evaluate the full SSOD model trained on Compcars and Cityscapes; its ablated versions with specific individual components removed (as described in Sec. 3.4.2 – 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 3.3). Similar to KITTI, for Cityscapes too, SSOD-Full achieves the best performance (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).

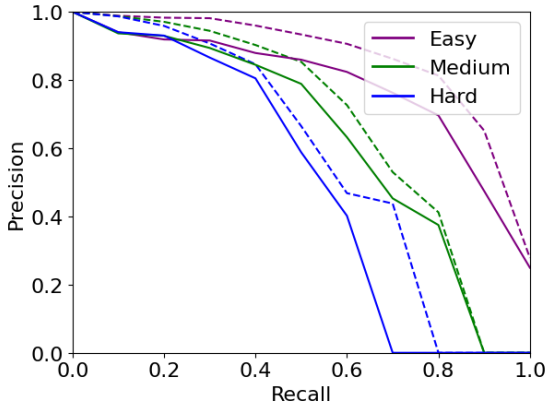


Figure 3.5. Precision-recall curves on KITTI. Curves for SSOD with IOU thresholds of 0.5 (bold lines) and 0.45 (dashed lines).

3.4.5 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. 3.5 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.

3.5 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.

Chapter 4

Learning Image Synthesis and Decomposition Through Mutual Supervision

4.1 Introduction

State-of-the-art sampling-based rendering engines (e.g., Mitsuba [81]) are able to generate photo-realistic images of virtual objects which are nearly indistinguishable from real-world photographs. However, this is not an easy task to accomplish since all intrinsic physical aspects of the virtual object must be accurately modeled, such as accurate 3D geometry, detailed textures and physically-based materials. While some of these intrinsics are abundant on the internet, such as the geometry of 3D objects (e.g. Turbosquid and 3D Warehouse), others are hard to obtain, such as high-quality materials – ideally in the form of a highly-accurate spatially-varying BRDF. In addition, sophisticated and slow rendering algorithms with many tunable parameters (lighting, environment map, camera model, post-processing) are required for turning 3D content into photo-realistic images. These parameters are often tuned individually with each rendered image, making it hard to create a large and diverse set of rendered images. On the other hand, obtaining a large number of real images which capture the complex interaction of light

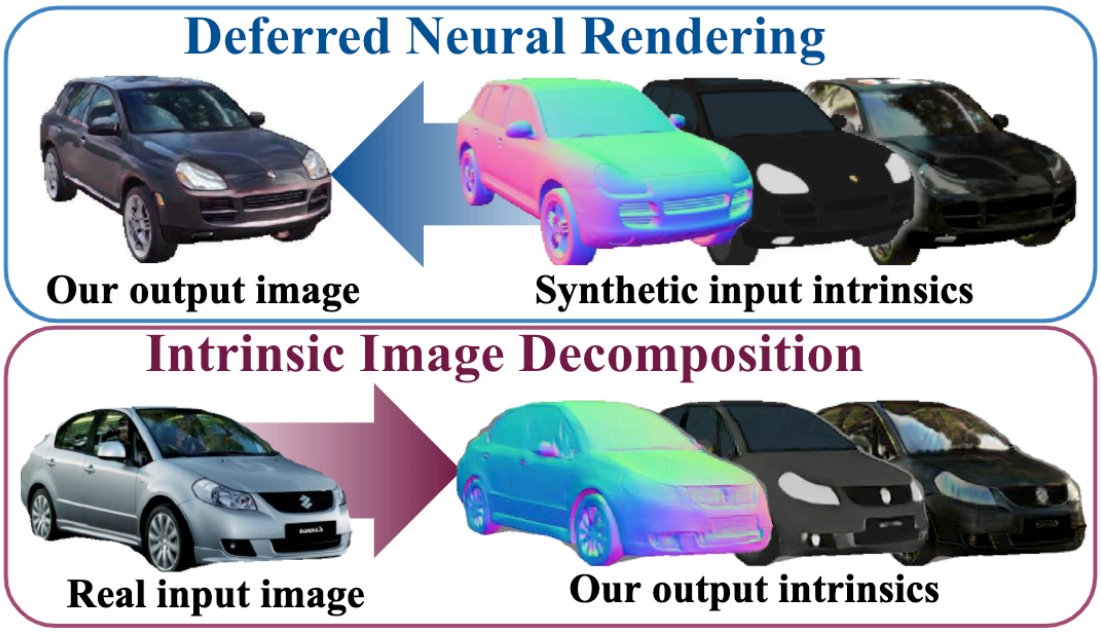


Figure 4.1. Deferred Neural Rendering and Intrinsic Image Decomposition. At training time, our model exploits normals, albedo and reflections from a small set of 3D models as well as a large set of *unpaired* RGB images of the same object category. Our model solves two tasks simultaneously: (i) generating photo-realistic images given the input geometry and basic intrinsic properties, and (ii) decomposing real images back into their intrinsic components.

with scene geometry and surface properties is easy. This makes the idea of learning neural image synthesis from real images very attractive.

Several works on conditional image generation [27, 77, 141, 179] have exploited paired datasets of real images with semantic information, including semantic segmentation [27, 141] and body part labels [105] for training realistic image synthesis models. However, such sparse inputs only allow limited control over the generated image. This limits the applicability of these methods, e.g., in virtual reality or video game simulations where precise control over the output is essential. Training a conditional image generation model from richer control inputs would require a large dataset of paired real images with pixel aligned intrinsic properties such as 3D structure, textures, materials and reflections. Obtaining such a dataset is hard in practice.

Our goal is to take a step towards learning a highly controllable realistic image

synthesis model without requiring real world images with aligned 3D models. Our key insight is that learning the inverse task of intrinsic decomposition is helpful for learning image synthesis from real images and vice-versa. We therefore train both, the forward rendering process and the reverse intrinsic decomposition process, jointly using a single objective as illustrated in Fig. 4.1. Inspired by recent results in unpaired image-to-image translation [73, 114, 200], we train our model using an small set of synthetic 3D models of an object category as well as a large unpaired dataset of real images of the same category.

Towards this goal, we exploit a technique from real-time rendering called *Deferred Rendering* which splits the rendering process into two stages and thus improves efficiency. In the first stage, the geometry of the scene along with its textures and material properties are projected onto a 2D pixel grid, resulting in a set of 2D intrinsic images which capture the geometry and appearance of the object. This step is efficient since it does not require physically accurate path tracing but relies on simple rendering operations. In the second "deferred" stage, lighting, shading and textural details are added to form the final rendered image. Our goal is to replace this second deferred stage of the rendering process with a neural network which we call *Deferred Neural Rendering* (DNR) network. To ensure that the input information is represented in the output image, we decompose it back into its intrinsics using a second Intrisic Image Decomposition (IID) network. However, we found that using this cycle alone leads to overfitting, especially in the IID network. To improve the IID network, we introduce a second *Decomposition cycle* in which we train the IID network to decompose real images.

Overall, our model follows a similar dual cycle training setup as proposed in [200] and [190]. However, an important conceptual difference to these works is that our task is not a one-to-one but a one-to-many mapping. Different realistic images can be generated from the same set of intrinsic maps as the intrinsics do not uniquely define the image. Likewise, a single image can be explained using different intrinsic decompositions due to projection from the higher dimensional intrinsics into the RGB image space.

We therefore introduce a shared adversarial discriminator between the input and the reconstruction at the end of each cycle. Our model enables both highly photo-realistic

image synthesis and accurate intrinsic image decomposition. We summarize our main contributions as follows:

- We propose the Intrinsic Autoencoder, a method to jointly train photo-realistic image synthesis and intrinsic image decomposition using cycle consistency losses without using any paired data.
- We propose a shared discriminator network that enables better generalization and proves key for learning both tasks without paired training data.
- We analyze the importance of various model components using quantitative metrics and human experiments. We also show that our method recovers accurate intrinsic maps from challenging real images.

4.2 Related Work

Differentiable Rendering. A standard way of synthesizing images from a given geometry and material is to use rendering engines. Several works try to implement the rendering process in a differentiable manner, amenable to neural networks. The work of [17] used differentiable rendering with deformable face models for face reconstruction. The works of [118] and [95] proposed rasterization-based differentiable renderers but only support local illumination. In order to support more realistic image formation, some other works [26, 50, 51, 107] propose to back-propagate though path tracing. Differentiable rasterizers are relatively fast, but at the same time highly restrictive as they do not support complex global illumination. While differentiable path tracers produce more realistic images, they are usually quite slow, thus restricting their usage to specific applications. Another drawback of differentiable renderers is that they require a detailed representation of the rendering input in terms of geometry, illumination, materials and viewpoint. In this work, we bypass the specification of complex image formation by training a CNN to directly generate realistic images from given geometry and material inputs.

Neural Image Synthesis. Generative models such as Generative Adversarial Networks [57] and Variational Auto-Encoders (VAE) [98] are widely use to synthesize realistic images from a latent code. In contrast, our goal is to perform conditional image synthesis which allows more fine-grained control over the image generation process. Some popular conditional image generation approaches are label-to-image translation [124, 136], image-to-image translation [27, 39, 77, 114, 179, 200] and text-to-image generation [71, 149, 184, 193]. Earlier works [27, 39, 77] on conditional image-to-image generation are mostly supervised with paired data from both domains. Several works [114, 200] propose a way to use unpaired data from both domains for conditional image generation. Other advances in conditional image generation include innovations in network architectures and loss functions for generating high resolution images [179] and generating multiple diverse images [31, 73, 74, 188]. In this work, we develop a model for photo-realistic geometry-to-image translation using only unpaired training data as supervision. Our work is closely related to [2] which also considers geometry-to-image translation, but requires paired training data. Our work belongs to the family of unpaired conditional image generation models with architecture and losses (e.g., shared discriminator) specialized for the geometry-to-image translation. Our model outperforms state-of-the-art unpaired image-to-image translation models [73, 200] by a large margin.

Intrinsic Image Decomposition is a long standing problem in computer vision. [9] poses the task as an optimization problem with a set of hand-crafted priors for shape, shading albedo etc. On the other hand supervised methods like [84, 163, 164] use synthetic data to train the model followed by refinement on real images. However, synthetic data might not capture all the real world statistics and models trained with synthetic data might now generalize well to real images. Recently, several self-supervised intrinsic decomposition methods have been proposed [111, 117, 119]. [117] uses single images during both training and inference stages. [111, 119] make use of multiview images or video sequences of the scene during training and infer on a single image . Our work falls in the realm of self-supervised intrinsic image decomposition. We do not use any paired synthetic data or multiview sequences to train out model. Instead, we rely on jointly training models for neural rendering and image decomposition.

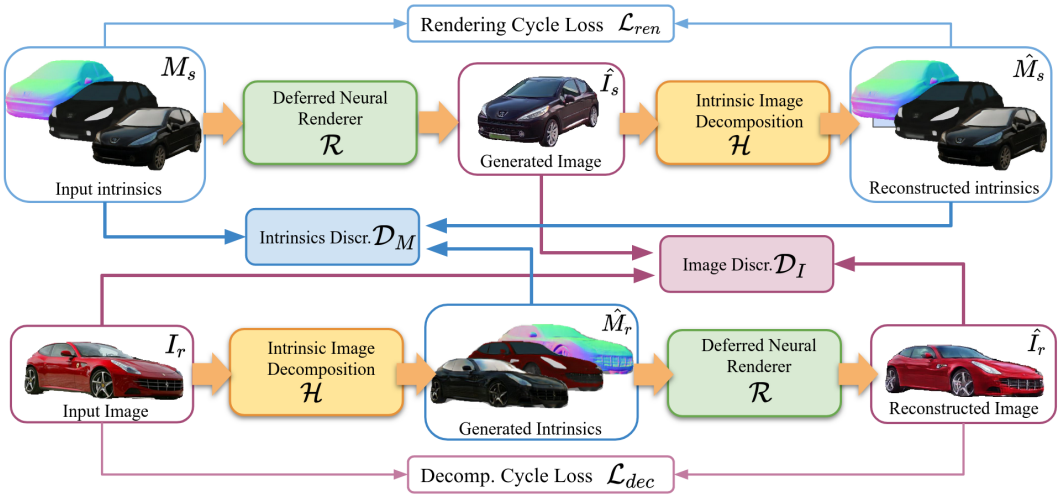


Figure 4.2. Intrinsic Autoencoder. Our model comprises two cycles: The first cycle (blue) auto-encodes a set of intrinsics rendered from 3D CAD models using appearance as latent representation. The second cycle (red) auto-encodes real images using image intrinsics as representation. Consistency is achieved through a combination of cycle losses and shared adversarial losses. Networks sharing the same weights are illustrated with the same color (green/yellow).

4.3 Method

Our Intrinsic Autoencoder model (Fig. 4.2) consists of two generator networks $\mathcal{R}$ and $\mathcal{H}$ for Deferred Neural Rendering and Intrinsic Image Decomposition, respectively. The Deferred Neural Rendering Network $\mathcal{R} : M \rightarrow \hat{I}$ takes as input a set of intrinsic maps $M = \{A, N, F\}$. The object's surface normal vectors in the view coordinate system $N \in \mathbb{R}^{H \times W \times 3}$ provides the Deferred Neural Rendering Network important information about the local shape of the object which is necessary for creating shading and reflection in the output image. The albedo $A \in \mathbb{R}^{H \times W \times 3}$ is a pixel-wise RGB value that describes the material or texture color at every pixel, ignoring any lighting effects. Finally, the environment reflections $F \in \mathbb{R}^{H \times W \times 3}$ are computed by projecting a high dynamic range environment map onto the 3D model. Note that this simple projection operation does not involve any complicated sampling or ray-tracing operations. As shown in our experiments, the Deferred Neural Renderer can also be trained with a subset of those inputs since it is able to compensate for the missing information. The DNR network

$\mathcal{R} : M \rightarrow \hat{I}$ transforms all the input intrinsics M into a realistic image $\hat{I} \in \mathbb{R}^{H \times W \times 3}$ that corresponds to the input intrinsics. Similarly, the Intrinsic Image Decomposition (IID) network $\mathcal{H} : I \rightarrow \hat{M}$ performs the opposite task by taking an input image I and predicting its intrinsics $\hat{M} \in \mathbb{R}^{H \times W \times 9}$.
Supervised training of $\mathcal{R}$ and $\mathcal{H}$ on real data is typically difficult due to the lack of real training image and intrinsics pairs (I_r, M_r). Instead, we use a combination of cycle-consistency losses and adversarial losses that require no paired training examples. This allows us to leverage a large dataset of real images $\{I_r^i\}_{i=0}^n$ and an unpaired set of synthetically generated intrinsic maps $\{M_s^i\}_{i=0}^m$. In the following, we detail our cycle consistency losses and the novel shared adversarial losses.

4.3.1 Cycle Consistency

Rendering Cycle. The goal of the rendering cycle is to train $\mathcal{R}$ in order to produce realistic images $\hat{I}_s = \mathcal{R}(M_s)$ from synthetic intrinsic maps M_s. To train $\mathcal{R}$ without paired data, we use the inverse transformation $\mathcal{H}$ which decomposes the predicted image $\hat{I}_s$ back into its intrinsic maps $\hat{M}_s = \mathcal{H}(\mathcal{R}(M_s))$ as illustrated in Fig. 4.2. We encourage consistency of the intrinsics using the rendering cycle consistency loss which is defined as the *Smooth*-L_1 distance between the input and reconstructed intrinsics

$$\mathcal{L}_{ren}(\mathcal{R}, \mathcal{H}, M_s) = \|\mathcal{H}(\mathcal{R}(M_s)) - M_s\|_1. \tag{4.1}$$

Decomposition Cycle. Similarly, we train $\mathcal{H}$ to generate intrinsic maps $\hat{M}_r = \mathcal{H}(I_r)$ from real images I_r. To ensure consistency with the input I_r, the output intrinsics $\hat{M}_r$ are passed to the deferred neural renderer $\mathcal{R}$ to reconstruct the image $\hat{I}_r = \mathcal{R}(\mathcal{H}(I_r))$. The decomposition cycle consistency loss is defined by:

$$\mathcal{L}_{dec}(\mathcal{R}, \mathcal{H}, I_r) = \|\mathcal{R}(\mathcal{H}(I_r)) - I_r\|_1. \tag{4.2}$$

The combined cycle consistency loss is then defined as:

$$\mathcal{L}_{cyc}(\mathcal{R}, \mathcal{H}, I_r, M_s) = \mathcal{L}_{ren}(\mathcal{R}, \mathcal{H}, M_s) + \mathcal{L}_{dec}(\mathcal{R}, \mathcal{H}, I_r) \tag{4.3}$$

To ensure that the predicted normals $\hat{N}_s = \mathcal{H}_N(I_r)$ and the reconstructed real normals $\hat{N}_r = \mathcal{H}_N(\mathcal{R}(M_s))$ are properly normalized, we exploit an additional normalization loss $\mathcal{L}_{\text{norm}}$:

$$\mathcal{L}_{\text{norm}}(\mathcal{R}, \mathcal{H}, I) = \mid 1 - \|\mathcal{H}_N(I_r)\|_2 \mid + \mid 1 - \|\mathcal{H}_N(\mathcal{R}(M_s))\|_2 \mid$$

4.3.2 Shared Adversarial Loss

While the cycle consistency loss ensures that the network input can be reconstructed from its output, it does not place any importance on the realism of that output. Additionally, the cycle consistency loss assumes a one-to-one deterministic mapping between the input and output. While this is a reasonable condition for some image-to-image translation tasks [200], it is violated when translating between images and their intrinsic properties. Decomposing an RGB image into its high-dimensional intrinsic properties is a one-to-many transformation since multiple decompositions can be consistent at the same time with the same image, e.g., a gray patch may correspond to a gray diffuse surface or a black glossy surface with specular highlight. Likewise, the process of creating an image from an incomplete set of intrinsic properties involves making additional predictions about missing attributes like lighting conditions, optical aberrations, noise or higher-order light interactions. To better capture this multi-modal relationship, we use an adversarial loss between the input and its reconstruction.

An adversarial discriminator $\mathcal{D}$ is a classification model trained to predict if a data sample is produced by a generative model or if it stems from the true data distribution. To train our Intrinsic Autoencoder, we use two adversarial discriminators, $\mathcal{D}_I$ for discriminating generated images $\hat{I}_{\{s,r\}}$ from real images I_r, and $\mathcal{D}_M$ for discriminating generated intrinsic maps $\hat{M}_{\{r,s\}}$ from synthetic intrinsic maps M_s. The discriminators help our model to learn the distribution of real images and synthetic intrinsics by optimizing the following adversarial [57] loss function

$$\mathcal{L}_{\text{adv}}(\mathcal{R}, \mathcal{H}, \mathcal{D}_I, \mathcal{D}_M) = \mathcal{L}^I_{\text{adv}}(\mathcal{R}, \mathcal{H}, \mathcal{D}_I) + \mathcal{L}^M_{\text{adv}}(\mathcal{R}, \mathcal{H}, \mathcal{D}_M) \tag{4.4}$$

where

$$\begin{aligned}\mathcal{L}^{I}_{\text{adv}}(\mathcal{R},\mathcal{H},\mathcal{D}_I) = \log(\mathcal{D}_I(I_r)) + \log(1-\mathcal{D}_I(\mathcal{R}(M_s)) \\ + \log(1-\mathcal{D}_I(\mathcal{R}(\mathcal{H}(I_r)))\end{aligned} \tag{4.5}$$

is our novel *shared adversarial image loss* which discriminates both between the real image I_r and the generated synthetic image $\hat{I}_s = \mathcal{R}(M_s)$, as well as between the real image I_r and the reconstructed real image $\hat{I}_r = \mathcal{R}(\mathcal{H}(I_r))$. Similarly, we define the *shared adversarial intrinsic loss* as

$$\begin{aligned}\mathcal{L}^{M}_{\text{adv}}(\mathcal{R},\mathcal{H},\mathcal{D}_M) = \log(\mathcal{D}_{\mathcal{M}}(M_s)) + \log(1-\mathcal{D}_M(\mathcal{H}(I_r)) \\ + \log(1-\mathcal{D}_M(\mathcal{H}(\mathcal{R}(M_s))))\end{aligned} \tag{4.6}$$

Using the reconstructed inputs $\hat{I}_r$ and $\hat{M}_s$ in addition to the generated samples $\hat{I}_s$ and $\hat{M}_r$ for training $\mathcal{D}_I$ and $\mathcal{D}_M$ makes the discriminators more robust and prevents overfitting. This is especially important when a relatively small number of 3D objects are used to create the synthetic intrinsic maps which can lead to a discriminator that recognizes the model features rather that the image realism.

4.3.3 Implementation and Training

We train our Intrinsic Autoencoder networks $\mathcal{R}, \mathcal{H}$ in addition to the adversarial discriminators $\mathcal{D}_M, \mathcal{D}_I$ from scratch by optimizing the joint objective

$$\min_{\mathcal{R},\mathcal{H}} \max_{\mathcal{D}_I,\mathcal{D}_M} \mathcal{L}_{\text{cyc}} + \mathcal{L}_{\text{norm}} + \mathcal{L}_{\text{adv}} \tag{4.7}$$

Our framework is implemented in PyTorch [144] and trained using Adam [97] with a learning rate of 0.0002. The Deferred Neural Rendering Network is a coarse-to-fine generator introduced in [179] for the deferred neural rendering network. The input to the network is of size 256×512 constructed by concatenating normals, albedo and reflections. The output of the network is an RGB image of size $256 \times 512 \times 3$. We use three networks $\mathcal{H} = \{\mathcal{H}_N, \mathcal{H}_A, \mathcal{H}_F\}$ for estimating the surface normals N, Albedo A and environment reflections F, respectively, from an image I. Each network has a ResNet architecture with 5 ResNet blocks.

Adversarial Discriminator Networks. Since the local structure of the generated images is mostly controlled by the input intrinsics, we want the image discriminator $\mathcal{D}_I$ to mainly focus on the global realism of the output. To address this, we use a multi-scale PatchGAN [179] discriminator which comprises two fully-convolutional networks that classify the local image patches at two scales, full and half resolution. The discriminator outputs a realism score for each patch instead of a single prediction per image. This has been shown to produce more detailed images for similar conditional image generation tasks [77, 179, 200]. The intrinsics discriminator $\mathcal{D}_M$ has the same architecture except that the input is a 9-channel stack combining all three intrinsic maps. We found that using a single discriminator for the combination of the intrinsic maps performs better than separate networks for each. This is likely due to the inter-dependence between the different intrinsic properties that allows the discriminator to detect inconsistencies between the generated intrinsic maps. We provide more architecture and training details in the supplementary material.

4.4 Experiments

Synthetic Data Generation. To generate the synthetic training data, we use dataset from [3] containing 28 3D car models covering 6 car categories (SUV, sedan, hatchback, station wagon, mini-van and van). Apart from the geometry, we do not need any physically-based materials or textures for the models. Instead, we assign to each car part a simple material with only two properties, the color and a scalar glossiness factor for computing reflection maps. We assign each 3D car part a fixed material from a set of 18 fixed materials. Additionally, we randomly pick one of 15 materials with different colors for the car body during the rendering process. Next, a camera position is randomly chosen within a radius of 8 meters and a maximum height of 3 meters. We use a fast OpenGL based rendering engine which operates at around 3 frames per second including the model loading time. It outputs the surface normals of the car model in the camera coordinate space and the albedo channels indicating the material color at each pixel without any lighting or shading. Finally, we produce the environmental reflections

by using a 360 degree environment map from [3]. These kind of reflections are very efficient to compute since they only require the view vector and the surface normal and do not rely on expensive path-tracing. We render 20,000 synthetic samples of normals, albedo and reflections.

Real training data. We obtain the real images from a fine grained car classification dataset presented in [99]. For convenience, we refer to this as the real car dataset. It contains 16,000 images of cars captured in various lighting conditions, resolutions and poses and with different camera sensors and lenses.

4.4.1 Baselines

Since our goal is to train with only unpaired data, we choose to benchmark our method against two state-of-the-art unpaired image generation approaches, CycleGAN[200] and MUNIT[73]. However, since both methods were originally designed for image-to-image translation rather than deferred rendering, we setup two additional strong baselines that highlight the importance of our contributions in improving the quality of our results.

CycleGAN and MUNIT. CycleGAN[200] is a generic method for translating between two domains without available paired data. MUNIT[73] aims at producing a diverse set of translations between different domains. We modify the two methods slightly to use our stacked 9 channel synthetic intrinsic maps as inputs.

Without shared discriminator. In this setup, we do not use the shared adversarial discriminator discussed in 4.3.2. Instead, we only use the discriminator $\mathcal{D}_I$ between generated image $\hat{I}_s$, real image I_r. Similarly, the discriminator $\mathcal{D}_M$ is used only between synthetic intrinsics M_s and generated intrinsics $\hat{M}_r$.

Only rendering cycle. Here, we train the model using only the deferred rendering cycle discussed in (Sec. 4.3.1) and do not use the decomposition cycle.

Figure 4.3. Images generated using our Deferred Neural Renderer. Inputs to the network are intrinsic maps consisting of albedo, normals and reflections, shown above the generated images.

4.4.2 Deferred Neural Rendering

To evaluate our approach for deferred neural rendering, we use the network $\mathcal{R}$ to produce images given synthetic intrinsic maps (albedo, normals, reflections) and compare it to other baselines, both qualitatively and quantitatively.

Qualitative results

Fig. 4.3 shows car images generated using our deferred neural renderer from the input synthetic intrinsic maps shown above them. The car models in the evaluation set have been previously seen by the generator, but the unique combination of pose and paint color has not been seen during training. Our approach is able to generate detailed photo-realistic images of cars with consistent geometry and distinct parts. We emphasize that the deferred neural rendering network is trained without any rendered or real geometry-image pairs. Instead, it is able to learn the appearance of different car parts from a large set of real car images. For more results see supplementary *

In Fig. 4.4 we compare the results of our full model to various baselines. The

*https://youtu.be/FOWoCeOAiug

Figure 4.4. Qualitative Comparison with baselines on Neural Rendering. Inputs to the network are intrinsic maps consisting of albedo, normals and reflections, shown above the generated images. Additional higher resolution results are provided in the supplementary materials.

results clearly show the improvements in visual quality achieved when using our full model. Specifically, MUNIT appears to be unable to preserve the geometry and albedo of the input in the generated image, CycleGAN images has significant artefacts on the windows, body, etc. When training our model without the shared discriminator, the resulting images suffer from irregular reflection patterns and a noisy image. This is

likely due to the strong overfitting required by the network to reproduce the input image exactly when using only an L_1 loss. The model trained without the decomposition cycle is not able to preserve the input intrinsics in terms of albedo and reflection.

In figure 4.5, we show the effect of input intrinsic maps on the quality of rendered images. When the model is trained only with normals as intrinsic input, the geometry of the result is well rendered but the color of different parts poorly defined. The model trained on both normals and albedo demonstrates sharper image quality but the hallucinated reflections by the network lacks lack realistic details. Finally, using the environmental reflections helps the network produce consistent and realistic images with sharp details.

Figure 4.5. Images generated using models trained with ablated inputs.

Quantitative results

We evaluate the quality of generated images using Fréchet Inception Distance(FID) [68] and Kernel Inception Distance(KID) [14]. Both metrics compute the distance between the features of two sets of images, obtained from a pre-trained CNN. Table 4.1 presents both the FID and KID between the images generated using various methods and the real images. Our full model achieves the lowest FID and KID values (47.6, 4.2) indicating that the rendered images from our model are closest to the distribution of real images compared to MUNIT [73] and CycleGAN [200]. Further, when we ablate each of the intrinsic map inputs, both FID and KID increase substantially. Notably, in the case of

	Cycle GAN	MUNIT	**Ours**	w/ Shared Discr	w/o Decom. Cyc.	w/o A	w/o N	w/o F
FID	103.3	99.0	**47.6**	59.2	99.6	88.7	60.2	56.7
KID	10.2	13.5	**4.2**	4.8	11.8	5.4	4.9	5.9

Table 4.1. FID and KID between real images and generated samples. All inputs are provided to the generator (Albedo, Normals and Reflections).

ablating albedo input, the highest increase in distances can be observed (88.7, 5.4), implying its importance for photo-realistic image generation. We conclude that albedo is the most important for our task followed by normals and reflections maps. In both cases where we ablate the decomposition cycle or rendering cycle, we observe a huge increase in the distances signifying the importance of using both cycle consistency losses during training. Finally, training with the setup of separate discriminators as mentioned in 4.4.1 leads to an increase in the distances.

Human Experiments

We design two experiments to measure the visual realism of generated car images using the Amazon Mechanical Turk platform to crowd source human evaluations. For each comparison, we presented 40 human subjects each with 50 image pairs to choose the more realistic looking image. The results are presented in Table 4.2. The first row presents experiments where one image is picked from the real images and the other is from one of the synthesis methods and presented in a random order. Images from our full model seem to be most confused with real car image since only 67.5% of choices were correct while in 32.5% of the trials the subjects choose our images to be the real one.

In the second experiment subjects are presented with an image generated by our full model and a matching image generated by one other synthesis methods. The results in the second row of Table 4.2 show that subjects choose our results to be more realistic over 80% of times when compare to CycleGAN and MUNIT. This clearly indicates a

	Cycle GAN	MUNIT	**Ours**	w/o Shared Discr.	w/o Decom Cyc.
Real Im	77.7%	75.6%	**67.5%**	68.9%	71.0%
Ours	80.0%	85.8%	–	57.6%	63.8%

Table 4.2. Human Subject Study. Comparisons to identify realistic images in an A/B test using Amazon Mechanical Turk. The numbers indicate the ratio of trials where the image from real or our model was chosen as more realistic compared to the image from the method on the header.

high level of visual quality of our generated images compared to those generated from existing methods. On the other hand, images from our ablated models appear to be much closer to our full model visual quality.

4.4.3 Intrinsic Image Decomposition

Qualitative results

In fig. 4.6, we show that the intrinsic decomposition network is able to decompose real car images into their intrinsic maps. We would like to emphasize that the model does not have access to ground-truth intrinsic maps for real images during the training phase. Also, these car models are not present in the synthetic training data.

Figure 4.7 compares the decompositions produced by our model to those from other baselines. Both CycleGAN [200] and MUNIT [73] show significant artificats and inconsistencies when trained to decompose real images. The USI3D [117] fails to generalize to real models since it was trained using synthetic data from ShapeNet [24]. Our model without decomposition cycle also recovers noisy albedo and normals due to overfit only to synthetic data. On the other hand, training without the shared discriminator leads to severe artefacts. This is because the rendering network tries to encode intrinsics information in the generated images in the form of high frequency artefacts such that the decomposition network can easily recover them.

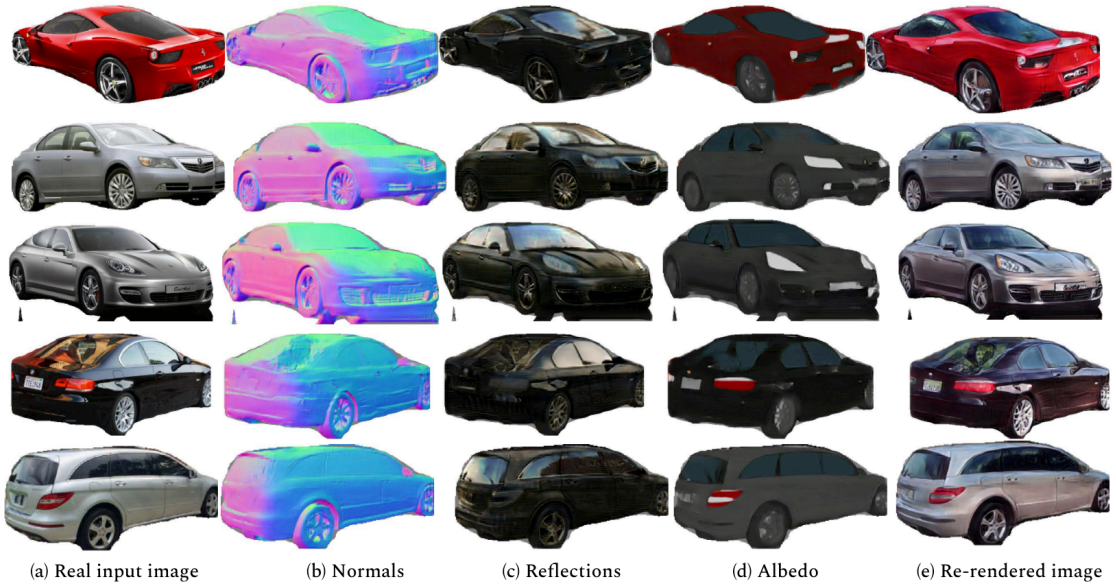


Figure 4.6. Results of our intrinsic decomposition network on real images. The first column shows the inputs to the network. Our model is able to decompose the sport car in first row accurately even though our synthetic training dataset does not include any sport cars at all. The car models of other inputs images are also not present in our synthetic dataset.

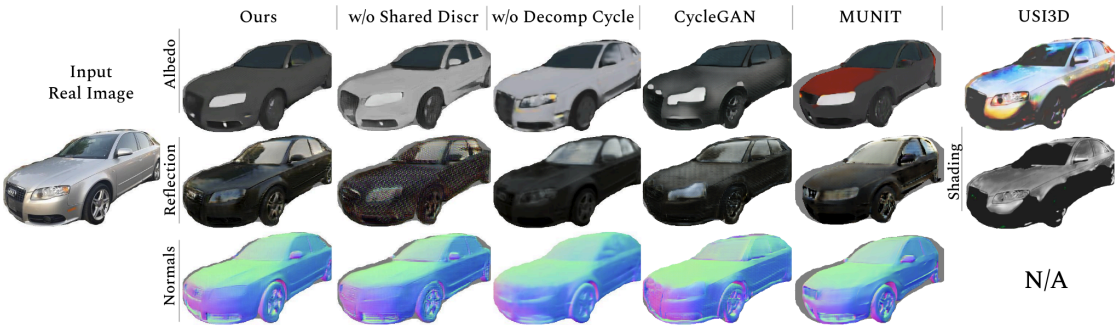


Figure 4.7. Comparison with Baselines for intrinsic decomposition. Note that USI3D [117] only produces albedo and shading and not reflections or normals.

Quantitative results

To evaluate the intrinsic maps predicted by the intrinsic image decomposition network ($\mathcal{H}$) we construct a synthetic dataset containing rendered RGB images and their corresponding intrinsic maps rendered using a standard Physically Based Renderer (Blender

	w/o Shared Discr.	w/o Decom. cycle	Cycle GAN	MUNIT	Ours
Normal Err.	17.75°	18.80°	27.82°	29.15°	**14.73**°
Albedo Err.	54.00	67.21	68.18	81.44	**52.74**
Reflection Err.	55.60	71.00	73.18	74.75	**51.74**

Table 4.3. Errors for the Intrinsic Decomposition Task. Our method achieves the lowest error on all tasks.

[18]). To obtain the error between predicted and ground truth normals, we compute the average cosine distance between them. The errors for albedo and reflection are the average ℓ_1 distances between the predicted and ground truth maps. Table 4.3 presents the errors of various methods for predicting intrinsic maps. Our full model has the least error for all the modalities followed by our model without the shared discriminator, without decomposition cycle and finally MUNIT and CycleGAN. This indicates that our model is able to learn accurate image decomposition while keeping generalization. Note that these PBR-rendered images have not been presented to our network during training.

4.4.4 Results on ShapeNet Airplanes

We train our model for the object class "Airplanes". We obtain the real images from FGVC-Aircraft dataset [121] which contains 10,000 images of airplanes. We use the 3D models of airplanes from the Shapenet dataset [24] to obtain our intrinsic maps. We follow the process mentioned in sec.4.4 to generate input training data. We use the normals and albedo as inputs to the network. Figure 4.8 illustrates realistic images generated using our deferred rendering network, demonstrating the ability of our method to handle low-quality mesh and texture models.

Figure 4.8. Images generated by our network trained on airplanes from ShapeNet [24].

4.5 Conclusion

In this chapter, we presented a joint approach for training a deferred rendering network for generating realistic images from synthetic image intrinsics and an intrinsic image decomposition network for decomposing real images of an object into its intrinsic properties. We trained the model using unpaired 3D models and real images. Our qualitative and quantitative experiments revealed that using a combination of shared adversarial losses and cycle consistency losses is able to produce images that are both realistic and consistent with the control input.

Chapter 5

Conclusion

In this thesis we explored the possibility of learning computer vision tasks like viewpoint estimation, object detection, image generation and decomposition using large collections of in-the-wild object images. In this chapter we present concluding remarks and future work based on these methods. We addressed self-supervised learning of some computer vision tasks as follows.

Viewpoint estimation We presented an approach for self-supervised learning of object viewpoint estimation only using large unlabeled collection of object images. We built a method based on the analysis-by-synthesis paradigm to jointly learn viewpoint prediction and viewpoint aware image synthesis using cycle consistency losses. We further utilized additional symmetry based constraints, style and viewpoint disentanglement based constraints to improve both the synthesis and viewpoint prediction networks. Our experiments show that we are able to achieve object viewpoint estimation accuracy better than other self-supervised approaches and competitive accuracy compared to supervised methods used for this task. This approach serves as a base line for future methods and also demonstrated that it is indeed possible to learn object viewpoint estimation only using unlabeled in-the-wild object images. The approach presents encouraging results for viewpoint prediction of un-occluded images. One possible future direction is to explore viewpoint estimation of occluded in-the-wild objects. This could be achieved by designing an occlusion and viewpoint aware synthesis network and concurrently a viewpoint

estimation network which can also reason about occlusions. This would significantly progress the field of object viewpoint estimation.

Object detection We presented an approach to learn object detection in a self-supervised manner only using collections of unlabeled images. To achieve this we utilized a controllable image synthesis network which used the input pose parameters of an object to generate an image with the object in the corresponding pose. The corresponding bounding box labels can be computed from the input pose parameters. This image label pair is in turn used to train the detector. We further designed several losses to adapt both the synthesis and detection network to new target domains. Experiments show that the method performs object detection better than existing self-supervised baselines and synthetic data based methods on average. Simultaneously, the method can also be adapted to new target data domains. 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. Going ahead, a direction to explore would be to take self-supervised object detection further to 3D object detection. A key step to achieve this is to improve the quality of composition of multiple objects, the image quality of the controllable synthesis network and its adherence to input pose parameters. Another improvement to the synthesis model is to be able to generate larger scene with multiple objects and better background. These improvements would provide high quality images and corresponding ground truth to train 3D object detection models in a self-supervised manner.

Image synthesis and decomposition We presented Intrinsic Autoencoders, a framework to learn the forward process of geometrically controllable image synthesis and the inverse process of image decomposition to obtain shape, albedo and reflections using unpaired images of objects and a small collection of 3D CAD models. Several supervised methods have been proposed for this task but obtaining a dataset where the images are exactly aligned with 3D CAD models is extremely difficult. We adopt a self-supervised learning paradigm where we train the image synthesis network and decomposition network jointly using cycle consistency losses and introduce a novel shared image discriminator to reduce image artifacts and improve the quality of generated images. Qualitative and quantitative analysis using human studies show that the images generated using the

geometric control are high quality, realistic and adhere to the inputs. Our image decomposition results are also significantly better than existing baselines. A prominent future direction for this approach is to model full scenes instead of objects to understand the geometry of surroundings. This could help us easily model our surroundings and generate novel views useful for AR/VR applications. Furthermore, networks can further be remodeled to handle deformable and articulated objects like humans, animals, etc. This would enable pose controlled generation of humans and self-supervised understanding of their geometry.

Bibliography

[1] "Scene understanding tasks in computer vision. image from teaching slides of cs231n course at stanford university." *cs231*, 2017. [Online]. Available: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf

[2] H. A. Alhaija, S. K. Mustikovela, A. Geiger, and C. Rother, "Geometric image synthesis," in *Asian Conference on Computer Vision (ACCV)*, 2018.

[3] H. A. Alhaija, S. K. Mustikovela, L. Mescheder, A. Geiger, and C. Rother, "Augmented reality meets computer vision: Efficient data generation for urban driving scenes," *International Journal of Computer Vision (IJCV)*, 2018.

[4] H. A. Alhaija, S. K. Mustikovela, L. M. Mescheder, A. Geiger, and C. Rother, "Augmented reality meets deep learning," in *British Machine Vision Conference (BMVC)*, 2017.

[5] H. A. Alhaija, S. K. Mustikovela, J. Thies, V. Jampani, M. Nießner, A. Geiger, and C. Rother, "Intrinsic autoencoders for joint deferred neural rendering and intrinsic image decomposition," in *International Conference on 3D Vision (3DV)*, 2020.

[6] P. Arbeláez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik, "Multiscale combinatorial grouping," in *CVPR*, 2014.

[7] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *ICML*, 2017.

[8] A. Avetisyan, M. Dahnert, A. Dai, M. Savva, A. X. Chang, and M. Nießner, "Scan2cad: Learning cad model alignment in rgb-d scans," 2018.

[9] J. T. Barron and J. Malik, "Shape, illumination, and reflectance from shading," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 8, 2015.

[10] A. Behl, O. Jafari, S. K. Mustikovela, H. A. Alhaija, C. Rother, and A. Geiger, "Bounding boxes, segmentations and object coordinates: How important is recognition for 3D scene flow estimation in autonomous driving scenarios?" in *International Conference on Computer Vision (ICCV)*, 2017.

[11] R. Benenson, S. Popov, and V. Ferrari, "Large-scale interactive object segmentation with human annotators," in *CVPR*, 2019.

[12] A. Bielski and P. Favaro, "Emergence of object segmentation in perturbed generative models," in *NeurIPS*, 2019.

[13] H. Bilen and A. Vedaldi, "Weakly supervised deep detection networks," in *CVPR*, 2016.

[14] M. Bińkowski, D. J. Sutherland, M. Arbel, and A. Gretton, "Demystifying MMD GANs," in *International Conference on Learning Representations (ICLR)*, 2018.

[15] M. Bińkowski, D. J. Sutherland, M. Arbel, and A. Gretton, "Demystifying MMD GANs," in *ICLR*, 2018.

[16] V. Blanz, T. Vetter *et al.*, "A morphable model for the synthesis of 3d faces." in *Siggraph*, 1999.

[17] ——, "A morphable model for the synthesis of 3D faces." in *Siggraph*, vol. 99, 1999.

[18] Blender Online Community, *Blender - a 3D Modelling and Rendering Package*. Blender Institute, Amsterdam: Blender Foundation, 2006.

[19] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," in *International Conference on Learning Representations (ICLR)*, 2019.

[20] A. Bulat and G. Tzimiropoulos, "How far are we from solving the 2d & 3d face alignment problem?" in *CVPR*, 2017.

[21] Y. Cabon, N. Murray, and M. Humenberger, "Virtual KITTI 2," *arXiv.org*, 2020.

[22] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," in *ECCV*, 2018.

[23] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," *CoRR*, vol. abs/2104.14294, 2021. [Online]. Available: https://arxiv.org/abs/2104.14294

[24] A. X. Chang, T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q.-X. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An Information-Rich 3D Model Repository," *arXiv*, vol. 1512.03012, 2015.

[25] F.-J. Chang, A. Tuan Tran, T. Hassner, I. Masi, R. Nevatia, and G. Medioni, "Faceposenet: Making a case for landmark-free face alignment," in *CVPR*, 2017.

[26] C. Che, F. Luan, S. Zhao, K. Bala, and I. Gkioulekas, "Inverse transport networks," *arXiv:1809.10820*, 2018.

[27] Q. Chen and V. Koltun, "Photographic Image Synthesis with Cascaded Refinement Networks," in *International Conference on Computer Vision (ICCV)*. Venice: IEEE, Oct. 2017.

[28] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton, "A simple framework for contrastive learning of visual representations," *CoRR*, vol. abs/2002.05709, 2020. [Online]. Available: https://arxiv.org/abs/2002.05709

[29] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in *NeurIPS*, 2016.

[30] X. Chen, J. Song, and O. Hilliges, "Monocular neural image based rendering with continuous view control," in *ICCV*, 2019.

[31] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[32] E. Collins, R. Achanta, and S. Susstrunk, "Deep feature factorization for concept discovery," in *ECCV*, 2018.

[33] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *CVPR*, 2016.

[34] ——, "The Cityscapes Dataset for Semantic Urban Scene Understanding," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[35] M. Cuturi, "Sinkhorn distances: Lightspeed computation of optimal transport," in *NeurIPS*, 2013.

[36] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.

[37] J. Devaranjan, A. Kar, and S. Fidler, "Meta-sim2: Learning to generate synthetic datasets," in *ECCV*, 2020.

[38] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial feature learning," *ICLR*, 2017.

[39] A. Dosovitskiy, J. T. Springenberg, M. Tatarchenko, and T. Brox, "Learning to generate chairs, tables and cars with convolutional networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, 2017.

[40] S. Ehrhardt, O. Groth, A. Monszpart, M. Engelcke, I. Posner, N. J. Mitra, and A. Vedaldi, "RELATE: Physically plausible multi-object scene synthesis using structured latent spaces," *NeurIPS*, 2020.

[41] G. Fanelli, M. Dantone, J. Gall, A. Fossati, and L. Van Gool, "Random forests for real time 3d face analysis," *IJCV*, 2013.

[42] Y. Feng, F. Wu, X. Shao, Y. Wang, and X. Zhou, "Joint 3d face reconstruction and dense alignment with position map regression network," in *ECCV*, 2018.

[43] R. M. Francisco Massa and M. Aubry, "Crafting a multi-task cnn for viewpoint estimation," in *BMVC*, 2016.

[44] N. Gählert, N. Jourdan, M. Cordts, U. Franke, and J. Denzler, "Cityscapes 3d: Dataset and benchmark for 9 dof vehicle detection," in *CVPR Workshops*, 2020.

[45] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual worlds as proxy for multi-object tracking analysis," in *CVPR*, 2016.

[46] Y. Gao, B. Liu, N. Guo, X. Ye, F. Wan, H. You, and D. Fan, "C-midn: Coupled multiple instance detection network with segmentation guidance for weakly supervised object detection," in *ICCV*, 2019.

[47] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, 2013.

[48] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *CVPR*, 2012.

[49] ——, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[50] I. Gkioulekas, A. Levin, and T. Zickler, "An evaluation of computational imaging techniques for heterogeneous inverse scattering," in *European Conference on Computer Vision (ECCV)*, 2016.

[51] I. Gkioulekas, S. Zhao, K. Bala, T. Zickler, and A. Levin, "Inverse volume rendering with material dictionaries," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 6, 2013.

[52] C. Godard, O. M. Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," *CoRR*, vol. abs/1609.03677, 2016. [Online]. Available: http://arxiv.org/abs/1609.03677

[53] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *CVPR*, 2017.

[54] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow, "Digging into self-supervised monocular depth prediction," October 2019.

[55] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NeurIPS*, 2014.

[56] ——, "Generative adversarial nets," in *NeurIPS*, 2014.

[57] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2014.

[58] A. Grabner, P. M. Roth, and V. Lepetit, "3d pose estimation and 3d model retrieval for objects in the wild," in *CVPR*, 2018.

[59] J. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. Á. Pires, Z. D. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko, "Bootstrap your own latent: A new approach to self-supervised learning," *CoRR*, vol. abs/2006.07733, 2020. [Online]. Available: https://arxiv.org/abs/2006.07733

[60] J. Gu, X. Yang, S. De Mello, and J. Kautz, "Dynamic facial analysis: From bayesian filtering to recurrent neural network," in *CVPR*, 2017.

[61] V. Guizilini, R. Ambrus, S. Pillai, and A. Gaidon, "Packnet-sfm: 3d packing for self-supervised monocular depth estimation," *CoRR*, vol. abs/1905.02693, 2019. [Online]. Available: http://arxiv.org/abs/1905.02693

[62] C. Gumeli, A. Dai, and M. Niebner, "ROCA: Robust CAD model retrieval and alignment from a single image," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2022. [Online]. Available: https://doi.org/10.1109%2Fcvpr52688.2022.00399

[63] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 9726–9735.

[64] P. Henderson and V. Ferrari, "Learning to generate and reconstruct 3d meshes with only 2d supervision," in *BMVC*, 2018.

[65] ——, "Learning single-image 3D reconstruction by generative modelling of shape, pose and shading," *IJCV*, 2019.

[66] P. Henderson, V. Tsiminaki, and C. Lampert, "Leveraging 2D data to learn textured 3D mesh generation," in *CVPR*, 2020.

[67] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local nash equilibrium," in *NeurIPS*, 2017.

[68] ——, "GANs trained by a two time-scale update rule converge to a local nash equilibrium," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2017.

[69] G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming auto-encoders," in *ICANN*, 2011.

[70] S. Honari, P. Molchanov, S. Tyree, P. Vincent, C. Pal, and J. Kautz, "Improving landmark localization with semi-supervised learning," in *CVPR*, 2018.

[71] S. Hong, D. Yang, J. Choi, and H. Lee, "Inferring semantic layout for hierarchical text-to-image synthesis," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[72] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *CVPR*, 2017.

[73] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz, "Multimodal unsupervised image-to-image translation," in *European Conference on Computer Vision (ECCV)*, 2018.

[74] L. Hui, X. Li, J. Chen, H. He, and J. Yang, "Unsupervised multi-domain image translation with domain-specific encoders/decoders," in *2018 24th International Conference on Pattern Recognition (ICPR)*, 2018.

[75] W.-C. Hung, V. Jampani, S. Liu, P. Molchanov, M.-H. Yang, and J. Kautz, "Scops: Self-supervised co-part segmentation," in *CVPR*, 2019.

[76] E. Insafutdinov and A. Dosovitskiy, "Unsupervised learning of shape and pose with differentiable point clouds," in *NeurIPS*, 2018.

[77] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[78] ——, "Image-to-image translation with conditional adversarial networks," *CVPR*, 2017.

[79] T. Jakab, A. Gupta, H. Bilen, and A. Vedaldi, "Unsupervised learning of object landmarks through conditional image generation," in *NeurIPS*, 2018.

[80] ——, "Learning landmarks from unaligned data using image translation," *CoRR*, vol. abs/1907.02055, 2019. [Online]. Available: http://arxiv.org/abs/1907.02055

[81] W. Jakob, *Mitsuba Renderer*, 2010.

[82] V. Jampani, S. Nowozin, M. Loper, and P. V. Gehler, "The informed sampler: A discriminative approach to bayesian inference in generative computer vision models," *Computer Vision and Image Understanding*, vol. 136, pp. 32–44, 2015.

[83] J. Janai, F. G"uney, A. Ranjan, M. J. Black, and A. Geiger, "Unsupervised learning of multi-frame optical flow with occlusions," in *European Conference on Computer Vision (ECCV)*, vol. Lecture Notes in Computer Science, vol 11220. Springer, Cham, Sep. 2018, pp. 713–731.

[84] M. Janner, J. Wu, T. Kulkarni, I. Yildirim, and J. B. Tenenbaum, "Self-supervised intrinsic image decomposition," in *Conference on Neural Information Processing Systems (NeurIPS)*, 2017.

[85] Z. Jie, Y. Wei, X. Jin, J. Feng, and W. Liu, "Deep self-taught learning for weakly supervised object localization," in *CVPR*, 2017.

[86] A. Kanazawa, S. Tulsiani, A. A. Efros, and J. Malik, "Learning category-specific mesh reconstruction from image collections," *CoRR*, vol. abs/1803.07549, 2018. [Online]. Available: http://arxiv.org/abs/1803.07549

[87] ——, "Learning category-specific mesh reconstruction from image collections," in *ECCV*, 2018.

[88] M. Kang, J. Shin, and J. Park, "Studiogan: A taxonomy and benchmark of gans for image synthesis," 2023.

[89] V. Kantorov, M. Oquab, M. Cho, and I. Laptev, "Contextlocnet: Context-aware deep network models for weakly supervised localization," in *ECCV*, 2016.

[90] A. Kar, A. Prakash, M.-Y. Liu, E. Cameracci, J. Yuan, M. Rusiniak, D. Acuna, A. Torralba, and S. Fidler, "Meta-sim: Learning to generate synthetic datasets," in *ICCV*, 2019.

[91] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," in *ICLR*, 2018.

[92] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *CVPR*, 2019.

[93] ——, "A style-based generator architecture for generative adversarial networks," in *CVPR*, 2019.

[94] H. Kato, Y. Ushiku, and T. Harada, "Neural 3d mesh renderer," in *CVPR*, 2018.

[95] ——, "Neural 3d mesh renderer," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[96] V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," in *CVPR*, 2014.

[97] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.

[98] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv:1312.6114*, 2013.

[99] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3D object representations for fine-grained categorization," in *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

[100] N. Kulkarni, A. Gupta, and S. Tulsiani, "Canonical surface mapping via geometric cycle consistency," in *ICCV*, 2019.

[101] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum, "Deep convolutional inverse graphics network," in *NeurIPS*, 2015.

[102] A. Kumar, A. Alavi, and R. Chellappa, "Kepler: keypoint and pose estimation of unconstrained faces by learning efficient h-cnn regressors," in *FG*, 2017.

[103] A. Kundu, Y. Li, and J. M. Rehg, "3d-rcnn: Instance-level 3d object reconstruction via render-and-compare," in *CVPR*, 2018.

[104] K. L Navaneet, P. Mandikal, V. Jampani, and V. Babu, "Differ: Moving beyond 3d reconstruction with differentiable feature rendering," in *CVPRW*, 2019.

[105] C. Lassner, G. Pons-Moll, and P. V. Gehler, "A generative model for people in clothing," in *International Conference on Computer Vision (ICCV)*, 2017.

[106] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o (n) solution to the pnp problem," *IJCV*, 2009.

[107] T.-M. Li, M. Aittala, F. Durand, and J. Lehtinen, "Differentiable monte carlo ray tracing through edge sampling," in *SIGGRAPH Asia 2018 Technical Papers*, 2018.

[108] X. Li, S. Liu, S. De Mello, K. Kim, X. Wang, M.-H. Yang, and J. Kautz, "Online adaptation for consistent mesh reconstruction in the wild," in *NeurIPS*, 2020.

[109] X. Li, S. Liu, K. Kim, S. De Mello, V. Jampani, M.-H. Yang, and J. Kautz, "Self-supervised single-view 3d reconstruction via semantic consistency," in *ECCV*, 2020.

[110] X. Li, S. Liu, K. Kim, S. D. Mello, V. Jampani, M. Yang, and J. Kautz, "Self-supervised single-view 3d reconstruction via semantic consistency," *CoRR*, vol. abs/2003.06473, 2020. [Online]. Available: https://arxiv.org/abs/2003.06473

[111] Z. Li and N. Snavely, "Learning intrinsic image decomposition from watching the world," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[112] S. Liao, E. Gavves, and C. G. M. Snoek, "Spherical regression: Learning viewpoints, surface normals and 3d rotations on n-spheres," in *CVPR*, 2019.

[113] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *CVPR*, 2017.

[114] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised Image-to-Image Translation Networks," *arXiv:1703.00848 [cs]*, Mar. 2017.

[115] P. Liu, M. R. Lyu, I. King, and J. Xu, "Selflow: Self-supervised learning of optical flow," *CoRR*, vol. abs/1904.09117, 2019. [Online]. Available: http://arxiv.org/abs/1904.09117

[116] S. Liu, T. Li, W. Chen, and H. Li, "Soft rasterizer: A differentiable renderer for image-based 3d reasoning," in *ICCV*, 2019.

[117] Y. Liu, S. You, Y. Li, and F. Lu, "Unsupervised learning for intrinsic image decomposition from a single image," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[118] M. M. Loper and M. J. Black, "OpenDR: An approximate differentiable renderer," in *European Conference on Computer Vision (ECCV)*, 2014.

[119] W.-C. Ma, H. Chu, B. Zhou, R. Urtasun, and A. Torralba, "Single image intrinsic decomposition without a single intrinsic image," in *European Conference on Computer Vision (ECCV)*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., 2018.

[120] S. Mahendran, H. Ali, and R. Vidal, "3d pose regression using convolutional neural networks," in *CVPR*, 2017.

[121] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi, "Fine-grained visual classification of aircraft," Tech. Rep., 2013.

[122] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *ECCV*, 2020.

[123] I. Misra and L. van der Maaten, "Self-supervised learning of pretext-invariant representations," *CoRR*, vol. abs/1912.01991, 2019. [Online]. Available: http://arxiv.org/abs/1912.01991

[124] T. Miyato and M. Koyama, "cGANs with projection discriminator," *arXiv:1802.05637*, 2018.

[125] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka, "3d bounding box estimation using deep learning and geometry," in *CVPR*, 2017.

[126] S. S. Mukherjee and N. M. Robertson, "Deep head pose: Gaze-direction estimation in multimodal video," *ACM MM*, 2015.

[127] S. K. Mustikovela, V. Jampani, S. De Mello, S. Liu, U. Iqbal, C. Rother, and J. Kautz, "Self-supervised viewpoint learning from image collections," in *CVPR*, 2020.

[128] K. L. Navaneet, A. Mathew, S. Kashyap, W.-C. Hung, V. Jampani, and R. V. Babu, "From image collections to point clouds with self-supervised shape and pose networks," in *CVPR*, 2020.

[129] G. Neuhold, T. Ollmann, S. Rota Bulò, and P. Kontschieder, "The mapillary vistas dataset for semantic understanding of street scenes," in *International Conference on Computer Vision (ICCV)*, 2017.

[130] T. Nguyen-Phuoc, C. Li, L. Theis, C. Richardt, and Y.-L. Yang, "HoloGAN: Unsupervised learning of 3D representations from natural images," in *International Conference on Computer Vision (ICCV)*, Nov. 2019.

[131] T. Nguyen-Phuoc, C. Li, L. Theis, C. Richardt, and Y.-L. Yang, "Hologan: Unsupervised learning of 3d representations from natural images," in *ICCV*, 2019.

[132] T. Nguyen-Phuoc, C. Richardt, L. Mai, Y.-L. Yang, and N. Mitra, "Blockgan: Learning 3d object-aware scene representations from unlabelled images," in *NeurIPS*, 2020.

[133] T. Nguyen-Phuoc, C. Richardt, L. Mai, Y.-L. Yang, and N. Mitra, "BlockGAN: Learning 3D object-aware scene representations from unlabelled images," in *Conference on Neural Information Processing Systems (NeurIPS)*, Nov. 2020.

[134] M. Niemeyer and A. Geiger, "Giraffe: Representing scenes as compositional generative neural feature fields," in *CVPR*, 2021.

[135] D. Novotny, D. Larlus, and A. Vedaldi, "Learning 3d object categories by looking around them," in *CVPR*, 2017, pp. 5218–5227.

[136] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier gans," in *International Conference on Machine Learning (ICML)*, 2017.

[137] D. P. Papadopoulos, J. R. Uijlings, F. Keller, and V. Ferrari, "We don't need no bounding-boxes: Training object class detectors using only human verification," in *CVPR*, 2016.

[138] ——, "Extreme clicking for efficient object annotation," in *CVPR*, 2017.

[139] T. Park, A. A. Efros, R. Zhang, and J.-Y. Zhu, "Contrastive learning for unpaired image-to-image translation," 2020.

[140] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu, "Semantic image synthesis with spatially-adaptive normalization," in *CVPR*, 2019.

[141] ——, "Semantic image synthesis with spatially-adaptive normalization," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[142] S. Park12, S. De Mello, P. Molchanov, U. Iqbal, O. Hilliges, and J. Kautz, "Few-shot adaptive gaze estimation," in *ICCV*, 2019.

[143] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NeurIPSW*, 2017.

[144] ——, "Automatic differentiation in PyTorch," in *NIPS-W*, 2017.

[145] G. Pavlakos, X. Zhou, A. Chan, K. G. Derpanis, and K. Daniilidis, "6-dof object pose from semantic keypoints," in *ICRA*, 2017.

[146] A. Prakash, S. Boochoon, M. Brophy, D. Acuna, E. Cameracci, G. State, O. Shapira, and S. Birchfield, "Structured domain randomization: Bridging the reality gap by context-aware synthetic data," *ICRA*, 2019.

[147] S. Prokudin, P. Gehler, and S. Nowozin, "Deep directional statistics: Pose estimation with uncertainty quantification," in *ECCV*, 2018.

[148] A. Ranjan, V. Jampani, K. Kim, D. Sun, J. Wulff, and M. J. Black, "Adversarial collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation," *CoRR*, vol. abs/1805.09806, 2018. [Online]. Available: http://arxiv.org/abs/1805.09806

[149] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," *arXiv:1605.05396*, 2016.

[150] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *NeurIPS*, 2015.

[151] Z. Ren, Z. Yu, X. Yang, M.-Y. Liu, Y. J. Lee, A. G. Schwing, and J. Kautz, "Instance-aware, context-focused, and memory-efficient weakly supervised object detection," in *CVPR*, 2020.

[152] Z. Ren, Z. Yu, X. Yang, M.-Y. Liu, A. G. Schwing, and J. Kautz, "Ufo^2: A unified framework towards omni-supervised object detection," in *ECCV*, 2020.

[153] H. Rhodin, M. Salzmann, and P. Fua, "Unsupervised geometry-aware representation for 3d human pose estimation," in *ECCV*, 2018.

[154] S. R. Richter, Z. Hayder, and V. Koltun, "Playing for benchmarks," in *ICCV*, 2017.

[155] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for Data: Ground Truth from Computer Games," in *European Conference on Computer Vision (ECCV)*, 2016.

[156] G. Riegler and V. Koltun, "Stable view synthesis," in *ECCV*, 2021.

[157] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[158] N. Ruiz, E. Chong, and J. M. Rehg, "Fine-grained head pose estimation without keypoints," in *CVPRW*, 2018.

[159] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic, "300 faces in-the-wild challenge: The first facial landmark localization challenge," in *ICCVW*, 2013.

[160] M. Sahasrabudhe, Z. Shu, E. Bartrum, R. Alp Guler, D. Samaras, and I. Kokkinos, "Lifting autoencoders: Unsupervised learning of a fully-disentangled 3d morphable model using deep non-rigid structure from motion," in *ICCVW*, 2019.

[161] K. Schwarz, Y. Liao, M. Niemeyer, and A. Geiger, "Graf: Generative radiance fields for 3d-aware image synthesis," in *NeurIPS*, 2020.

[162] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *ICCV*, 2017.

[163] S. Sengupta, J. Gu, K. Kim, G. Liu, D. Jacobs, and J. Kautz, "Neural Inverse Rendering of an Indoor Scene From a Single Image," in *International Conference on Computer Vision (ICCV)*. Seoul, Korea (South): IEEE, Oct. 2019.

[164] S. Sengupta, A. Kanazawa, C. D. Castillo, and D. W. Jacobs, "SfSNet: Learning shape, refectance and illuminance of faces in the wild," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[165] Y. Shen, R. Ji, S. Zhang, W. Zuo, and Y. Wang, "Generative adversarial learning towards fast weakly supervised detection," in *CVPR*, 2018.

[166] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[167] V. Sitzmann, J. Thies, F. Heide, M. Nießner, G. Wetzstein, and M. Zollhofer, "Deepvoxels: Learning persistent 3d feature embeddings," in *CVPR*, 2019.

[168] S. Song, S. Lichtenberg, and J. Xiao, "Sun rgb-d: A rgb-d scene understanding benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[169] H. Su, C. R. Qi, Y. Li, and L. J. Guibas, "Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views," in *CVPR*, 2015.

[170] Y. Sun, X. Wang, and X. Tang, "Deep convolutional network cascade for facial point detection," in *CVPR*, 2013.

[171] S. Suwajanakorn, N. Snavely, J. Tompson, and M. Norouzi, "Discovery of latent 3D key-points via end-to-end geometric reasoning," in *NeurIPS*, 2018.

[172] P. Tang, X. Wang, S. Bai, W. Shen, X. Bai, W. Liu, and A. Yuille, "PCL: Proposal cluster learning for weakly supervised object detection," *TPAMI*, 2018.

[173] A. Tewari, M. Zollhöfer, P. Garrido, F. Bernard, H. Kim, P. Pérez, and C. Theobalt, "Self-supervised multi-level face model learning for monocular reconstruction at over 250 hz," in *CVPR*, 2018.

[174] J. Thewlis, H. Bilen, and A. Vedaldi, "Unsupervised learning of object landmarks by factorized spatial embeddings," in *ICCV*, 2017.

[175] L. Tran and X. Liu, "Nonlinear 3d face morphable model," in *CVPR*, 2018.

[176] H.-Y. Tseng, S. De Mello, J. Tremblay, S. Liu, S. Birchfield, M.-H. Yang, and J. Kautz, "Few-shot viewpoint estimation," in *BMVC*, 2019.

[177] S. Tulsiani and J. Malik, "Viewpoints and keypoints," in *CVPR*, 2015.

[178] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders, "Selective search for object recognition," *IJCV*, 2013. [Online]. Available: http://www.huppelen.nl/publications/selectiveSearchDraft.pdf

[179] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional GANs," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[180] O. Wiles and A. Zisserman, "Silnet: Single- and multi-view reconstruction by learning from silhouettes," in *BMVC*, 2017.

[181] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes," in *RSS*, 2018.

[182] Y. Xiang, W. Kim, W. Chen, J. Ji, C. Choy, H. Su, R. Mottaghi, L. Guibas, and S. Savarese, "ObjectNet3D: A large scale database for 3D object recognition," in *ECCV*, 2016.

[183] Y. Xiang, R. Mottaghi, and S. Savarese, "Beyond pascal: A benchmark for 3d object detection in the wild," in *WACV*, 2014.

[184] T. Xu, P. Zhang, Q. Huang, H. Zhang, Z. Gan, X. Huang, and X. He, "Attngan: Fine-grained text to image generation with attentional generative adversarial networks," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[185] L. Yang, P. Luo, C. Change Loy, and X. Tang, "A large-scale car dataset for fine-grained categorization and verification," in *CVPR*, 2015.

[186] T.-Y. Yang, Y.-T. Chen, Y.-Y. Lin, and Y.-Y. Chuang, "Fsa-net: Learning fine-grained structure aggregation for head pose estimation from a single image," in *CVPR*, 2019.

[187] T.-Y. Yang, Y.-H. Huang, Y.-Y. Lin, P.-C. Hsiu, and Y.-Y. Chuang, "Ssr-net: A compact soft stagewise regression network for age estimation." *IJCAI*, 2018.

[188] Z. Yang, H. Liu, and D. Cai, "On the diversity of realistic image synthesis," *arXiv:1712.07329*, 2017.

[189] S. Yao, T. M. Hsu, J.-Y. Zhu, J. Wu, A. Torralba, B. Freeman, and J. Tenenbaum, "3d-aware scene manipulation via inverse graphics," in *NeurIPS*, 2018.

[190] Z. Yi, H. Zhang, P. Tan, and M. Gong, "DualGAN: Unsupervised dual learning for image-to-image translation," *CoRR*, vol. abs/1704.02510, 2017.

[191] A. Yuille and D. Kersten, "Vision as bayesian inference: analysis by synthesis?" *Trends in cognitive sciences*, 2006.

[192] Z. Zeng, B. Liu, J. Fu, H. Chao, and L. Zhang, "Wsod2: Learning bottom-up and top-down objectness distillation for weakly-supervised object detection," in *ICCV*, 2019.

[193] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. N. Metaxas, "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks," in *International Conference on Computer Vision (ICCV)*, 2017.

[194] K. Zhang, G. Riegler, N. Snavely, and V. Koltun, "Nerf++: Analyzing and improving neural radiance fields," *arXiv.org*, 2020.

[195] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *SPL*, 2016.

[196] R. Zhang, P. Isola, and A. A. Efros, "Split-brain autoencoders: Unsupervised learning by cross-channel prediction," in *CVPR*, 2017.

[197] X. Zhang, J. Feng, H. Xiong, and Q. Tian, "Zigzag learning for weakly supervised object detection." in *CVPR*, 2018.

[198] Y. Zhang, Y. Guo, Y. Jin, Y. Luo, Z. He, and H. Lee, "Unsupervised discovery of object landmarks as structural representations," in *CVPR*, 2018.

[199] X. Zhou, A. Karpur, L. Luo, and Q. Huang, "Starmap for category-agnostic keypoint and viewpoint estimation," in *ECCV*, 2018.

[200] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *International Conference on Computer Vision (ICCV)*, 2017.

[201] J.-Y. Zhu, Z. Zhang, C. Zhang, J. Wu, A. Torralba, J. Tenenbaum, and B. Freeman, "Visual object networks: image generation with disentangled 3d representations," in *NeurIPS*, 2018.

[202] X. Zhu, X. Liu, Z. Lei, and S. Z. Li, "Face alignment in full pose range: A 3d total solution," *PAMI*, 2017.

[203] L. Zitnick and P. Dollar, "Edge boxes: Locating object proposals from edges," in *ECCV*, 2014.