ABSTRACT

Title of dissertation: Neural Rendering Techniques for Photo-realistic Image Generation and Novel View Synthesis

Moustafa Mahmoud Meshry
Doctor of Philosophy, 2022

Dissertation directed by: Professor Abhinav Shrivastava
Department of Computer Science

Professor Larry S. Davis
Department of Computer Science

Recent advances in deep generative models have enabled computers to imagine and generate fictional images from any given distribution of images. Techniques like Generative Adversarial Networks (GANs) and image-to-image (I2I) translation can generate images by mapping a random noise or an input image (*e.g.*, a sketch or a semantic map) to photo-realistic images. However, there are still plenty of challenges regarding training such models and improving their output quality and diversity. Furthermore, to harness this imaginative and generative power for solving real-world applications, we need to be able to control different aspects of the rendering process; for example to specify the content and/or style of generated images, camera pose, lighting, . . . etc.

One challenge to training image generation models is the multi-modal nature of image synthesis. An image with a specific content, such as a cat or a car, can be generated with countless

choices of different styles (*e.g.*, colors, lighting, and local texture details). To enable user control over the generated style, previous works train multi-modal I2I translation networks, but they suffer from a complicated and slow training, and their training is specific to one target image domain. We address this limitation and propose a style pre-training strategy that generalizes across many image domains, improves the training stability and speed, and improves the performance in terms of output quality and diversity.

Another challenge to GANs and I2I translation is to provide 3D control over the rendering process. For example, applications such as AR/VR, virtual tours and telepresence require generating consistent images or videos of 3D environments. However, GANs and I2I translation mainly operate in 2D, which limits their use for such applications. To address this limitation, we propose to condition image synthesis on coarse geometric proxies (*e.g.*, a point cloud, a coarse mesh, or a voxel grid), and we augment these rough proxies with machine learned components to fix and compliment their artifacts and render photo-realistic images. We apply our proposal to solve the task of novel view synthesis under different challenging settings, and show photo-realistic novel views of complex scenes with multiple objects, tourist landmarks under different appearances, and human subjects under novel head poses and facial expressions.

Neural Rendering Techniques for Photo-realistic Image Generation and Novel View Synthesis

by

Moustafa Mahmoud Meshry

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2022

Advisory Committee:
Professor Abhinav Shrivastava, Chair/Advisor
Professor Larry S Davis, Co-Chair/Advisor
Professor David Jacobs
Professor Tom Goldstein
Professor Carol Espy-Wilson

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Motivation

The human brain processes a staggering amount of visual information on a daily basis. Through visual perception, we are able to understand the environment around us, anticipate future events and plan our actions. Imagine a simple scenario like walking down the street. We can easily interpret the scene around us; recognize static objects like buildings and trees, moving objects like cars and people, and even reason about missing information, like how the occluded part of a building looks behind that tree, or how that parked car looks like in 3D even though we only observe it from one side. It is therefore not surprising that more than 50 percent of our brain cortex is devoted to processing visual information. Complex as it is, human interaction with the visual world happens at ease and in a split of a second. We have this remarkable ability to digest visual inputs and extract useful concepts and information to interact with our surroundings.

1.1.1 Visual understanding (visuals $\rightarrow$ concepts)

The field of *Computer Vision* aims to bring human-like visual capabilities to computers. For many years, Computer Vision researchers focused on breaking down how humans digest and understand the visual world into tasks, and aimed to replicate each of those tasks. For example,

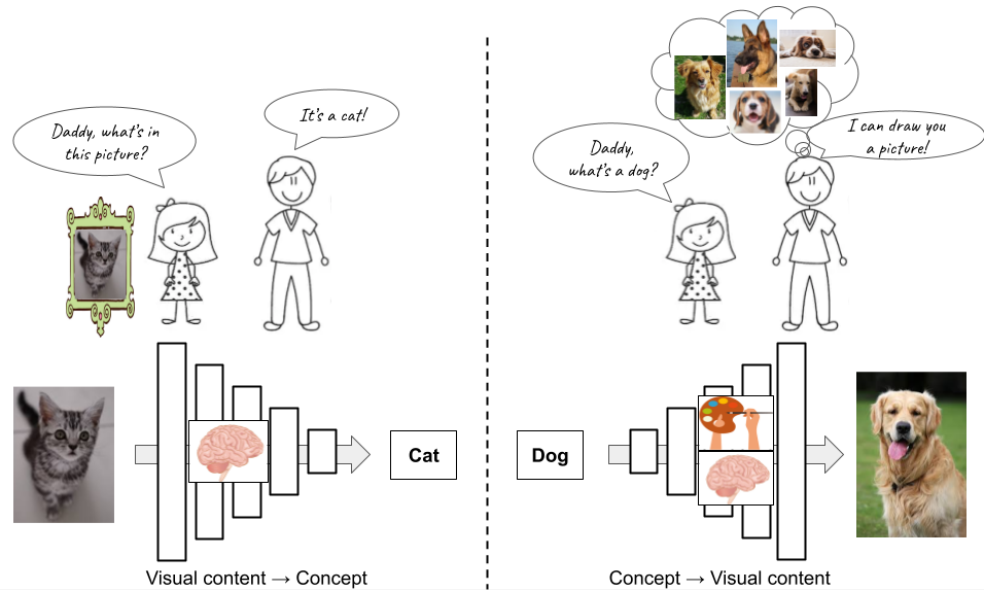


Figure 1.1: Human interaction with the visual world both consumes and creates visual content.

tasks like object detection, action recognition, video prediction, motion planning and many more have all been studied. With the emergence of *Deep Learning* in 2012, Computer Vision has experienced revolutionary advances at all fronts. An image or a video is fed to a deep neural network which extracts useful features. Those features encode abstract concepts and information such as action types for action recognition, object classes and bounding boxes for object detection, or the locations of body joints for human pose understanding and prediction.

1.1.2 Visual content creation (concepts → visuals)

The human interaction with the visual world is however not just about interpreting visual inputs and extracting useful information from it. It also goes in the reverse direction, where it starts from an abstract concept or a description, and creates a mental picture or a visualization (*e.g.*, Figure 1.1). For example, if we think about an abstract term like "dogs", our minds recall an entire distribution that falls under the umbrella of that concept. We can easily imagine countless

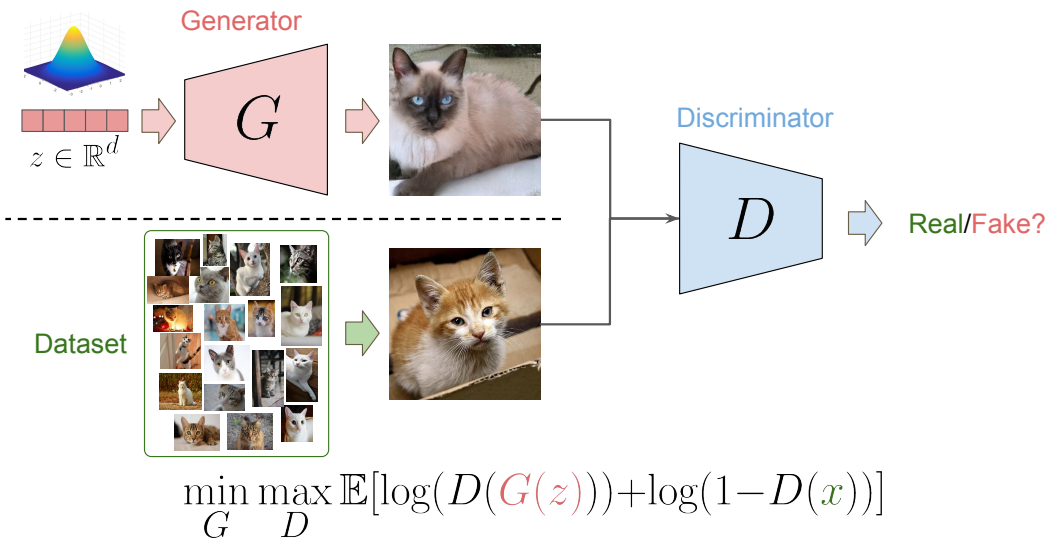


Figure 1.2: Training overview of *Generative Adversarial Networks* (GANs) [5].

visual realizations of that concept, like dogs of different breeds, or with different colors or poses. If we specify more details, like "A golden retriever standing on grass", then a mental picture is immediately drawn into our mind. This amazing gift of imagination is an essential part of our visual experience and perception.

Imagining and generating visual content is a hard skill/task, even for humans. This is evidenced by the small portion of people with the creative imagination and/or the skill to turn this imagination into physical art (*i.e.*, generate visual content). So, we could only imagine how difficult it would be to teach machines the power of imagination . . . or so we thought! For a long time, Computer Vision research has focused on discriminative learning tasks like object recognition and detection, but not as much on generative modeling where a machine would imagine and create new visual content. This however changed with the introduction of *Generative Adversarial Networks* [5] in 2014.

Generative Adversarial Networks (GANs) have had a significant impact on the fields of distribution learning and photo-realistic image and video synthesis. In its unconditional setting

(Figure 1.2), GANs learn a function, $G(z) \rightarrow I$, that maps a source distribution (typically a unit Gaussian) to a target distribution of real images. And such as we can sample real images from the target dataset, we can also sample continuous random codes, $z \in \mathbb{R}^d$, from the prior distribution, pass them through a generator $G(.)$ to synthesis an image I. The generator aims to synthesize images that fool a discriminator network D, which in turn learns to discriminate between real and fake images. Both the generator and the discriminator are trained jointly by playing a mini-max game against each other, where the generated images converge to the manifold of the target distribution as the training progresses. After the training has concluded, the network can imagine and create novel images from the target distribution by passing a randomly sampled code z through the generator network G. State-of-the-art works [11, 12, 13, 14, 15] can generate highly realistic images from various image categories, such as human faces, cars, bedrooms, and many of the ImageNet [16] classes.

Having machines imagine random images is a breakthrough. However, to harness the true power of this imaginative and generative capability, a form of user control needs to be introduced.

1.2 Background: Controllable photo-realistic image synthesis

Using computers to create useful visual content requires both controllability and photo-realism; we need to be able to control different aspects of the generated image (*e.g.*, camera pose, lighting, and texture), and the output needs to be of high quality and of a high degree of realism. In the following, we briefly review tools for photo-realistic rendering. This includes traditional techniques from Computer Graphics (§ 1.2.1), as well as more recent techniques from generative Machine Learning and Computer Vision (§§ 1.2.2 and 1.2.3).

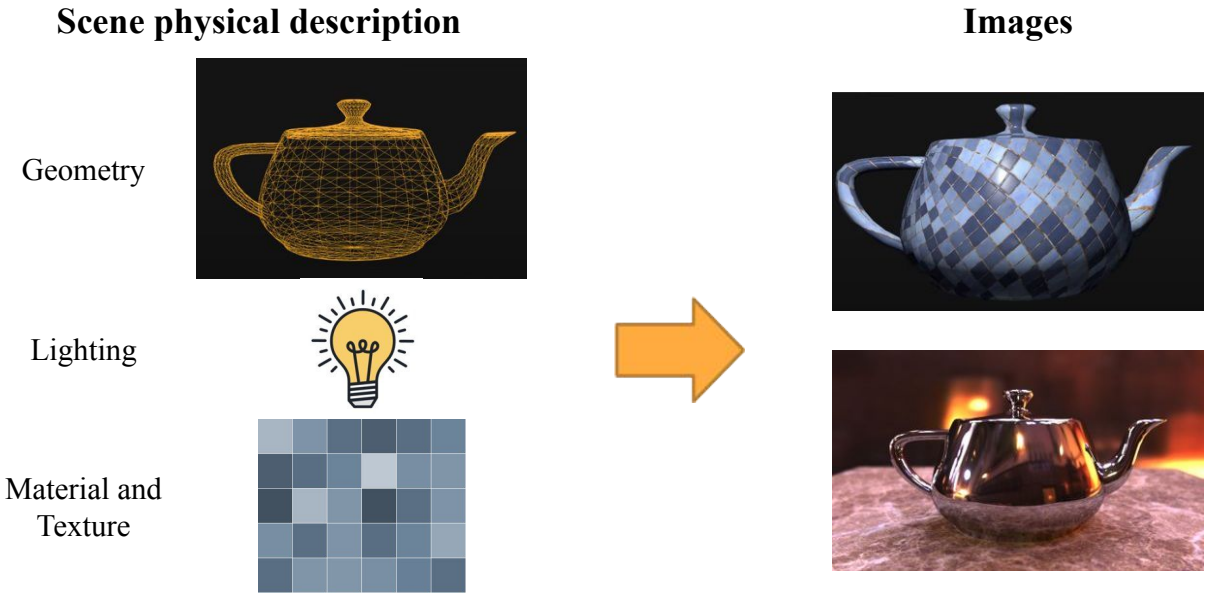
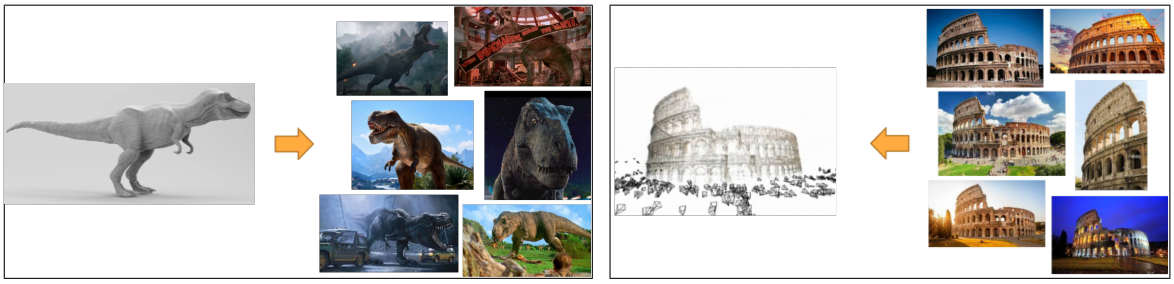


Figure 1.3: Given a physical description of a scene, Computer Graphics (CG) techniques simulate light transport and render photo-realistic images.

1.2.1 Computer Graphics (CG)

Photo-realistic image synthesis and rendering is a well studied and understood problem in Computer Graphics (CG). Given a physical description of the scene such as the geometry, light sources illuminating the scene, and surface materials and texture, Computer Graphics techniques, such as ray tracing, can simulate light transport and render photo-realistic images of the scene (*e.g.*, Figure 1.3).

However, this requires expensive manual work to obtain or build the 3D geometry of different objects in the scene, and artists need to set up the entire scene from the ground up. Assuming we can obtain this really good geometry, then Computer Graphics enables full control on how to render it, and we can generate a variety of photo-realistic images (*e.g.*, Figure 1.4a). To avoid or reduce the amount of expensive manual work required for geometric scene modeling, we can try to retrieve the geometry of a scene from a set of images. This is the "3D Reconstruction" problem



(a) Photo-realistic rendering (Computer Graphics). (b) 3D Reconstruction (Computer Vision).

Figure 1.4: Computer Graphics requires high quality geometry to render photo-realistic images. Standard 3D reconstruction techniques can recover geometry from images, but the recovered geometry lacks in quality.

which is well studied in Computer Vision. For example, Structure from Motion (SfM) [17, 18, 19] and Multi-View Stereo (MVS) [20, 21] techniques can process thousands of images and recover a dense point cloud representation of the scene. But, if render the recovered geometry, then it does not look very nice (Figure 1.4b). Therefore, photo-realistic image synthesis remained limited to Computer Graphics for a long time; that is until GANs came out.

1.2.2 Generative Machine Learning

Unconditional GANs can generate random photo-realistic images from a target distribution. However, one main limitation is that it is very tricky to control the synthesis process of unconditional GANs. For example instead of using GANs to generate random 'dog' images, we might want to specify certain aspects like breed, scale, pose, color, ... etc. In 2017, Isola *et al.* [7] paved the way for such user control by proposing an image-to-image (I2I) translation framework, where the synthesis process is conditioned on an input image that describes certain attributes of the target output. Therefore, image-to-image (I2I) translation learns to map images from a source domain A to a target domain B. For example, semantic maps $\rightarrow$ scenes or sketches $\rightarrow$ photo-realistic images. Many problems in Computer Vision and Graphics can be cast as I2I translation,

Table 1.1: Pros and cons of Computer Graphics *vs.* Generative Models for photo-realistic image synthesis.

Computer Graphics (CG)	**Generative Models (ML/CV)**
✓ Explicit control over image synthesis.	✓ End-to-end training (no manual work).
✓ Operates in 3D.	✓ Fast rendering (feed forward).
✗ Expensive manual work from artists to build/obtain high quality geometry and setup the entire scene.	✗ Operates in 2D.
✗ Slow rendering for high quality synthesis.	✗ No fine-grain control over the synthesis process.
	✗ Needs lots of training data.

such as inpainting [22], colorization [23, 24], super-resolution [25], image de-noising [26], rendering [27, 28, 29], and many more [30, 31, 32]. While I2I translation provides some degree of control over the image synthesis process of GANs, it is still far from the fine-grain level of control as in Computer Graphics.

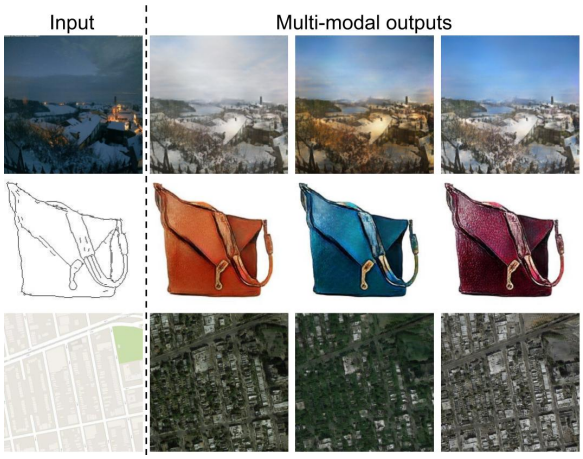
1.2.3 Neural Rendering

In the previous sections, we discussed two alternatives for photo-realistic image synthesis. On one hand Computer Graphics gives full control over the synthesis process, including camera placement and defining light sources and material properties of different objects. It also enables explicit manipulation of the 3D geometry. However, CG requires obtaining or building high quality assets, as well as expensive manual work from artists for geometric modeling, setting up the scene, and defining lighting and surface properties.

On the other hand, generative models and I2I translation frameworks can be trained end-to-end and can be relatively faster to render, since rendering only requires a simple feed forward pass through a neural network. But on the downside, they mainly operate in 2D, they lack the fine-grain control compared to CG, and they require large training data. Table 1.1 contrasts the pros

and cons of of image synthesis using Computer Graphics *vs.* generative modeling, which suggests that both fields can compliment many of the limitations of each others. Therefore, it was intuitive to combine the best of both worlds, giving rise to the sub-filed of *Neural Rendering* for photo-realistic image and video synthesis. The key idea of Neural Rendering is to make the forward rendering process of CG differentiable. Then we can start from some scene representation (*e.g.*, parametric models such as a face or body mesh [33, 34]), render images using the differentiable rendering process, compute a loss in the image space, and then backpropagate the loss to improve the initial representation (*e.g.*, the parameters of the parametric model). Even better, since the rendering process is now differentiable, then learnable components can be added either in the rendering part or in the representation or in both. This allows relaxing the CG requirement for having high quality geometric assets by instead constructing a coarse geometric proxy (*e.g.*, a point cloud, a coarse mesh, or a coarse voxel grid), and augmenting this coarse representation with machine learned components to compliment and fix their artifacts and render photo-realistic images.

Since I2I translation and Neural Rendering are relatively new fields, there are plenty of open challenges to be tackled. Two distinct directions are (1) improving the training of GANs and I2I translation networks, and (2) expanding their applications to solve real-world problems. Therefore, this dissertation focuses on those two directions by addressing some of the training challenges of I2I translation networks (§ 1.3), while also expanding their applications, specifically to the task of Novel View Synthesis (§ 1.4).



(a) Style multi-modality in I2I translation.

(a) Generated image (b) Stochastic variation (c) Standard deviation

(b) Local stochastic variation between multiple generations of StyleGAN [13]. Figure adopted from [13].

Figure 1.5: Example challenges of training I2I translation networks. Left: I2I is multi-modal in nature; the same input can be mapped to multiple plausible outputs. Right: Examples of many possible outputs with stochastic variations that all satisfy the GAN objective. Applying reconstruction losses penalizes such variations, which misleads I2I training.

1.3 Addressing challenges of I2I training

Training unconditional GANs is known to be difficult and unstable. For example "mode collapse" is a major concern when training GANs, where the generator collapses to synthesizing one or few images that fool a discriminator network, and therefore fails to accurately represent the variability of the target distribution. Also, the training of GANs is very slow and the output can suffer from visible artifacts. Although several works have proposed improvements to stabilize GAN training and allow for higher quality image synthesis [11, 14, 15, 35, 36], improving GAN training remains an open research problem.

I2I translation inherits several of the training challenges of unconditional GANs, while also introducing new training challenges specific to its image-conditional setup. For example, translating a sketch of a handbag to a photo-realistic image is a multi-modal problem; the same bag sketch can be mapped to bags with different colors and textures (*e.g.*, Figure 1.5a). Also, there

Figure 1.6: Novel View Synthesis is the task of generating images of a scene from novel unobserved views, given only a sparse set of observations of the scene. (Figure adapted from [6]).

are infinite possible local variations in the generated image that add realism without changing the essence of the image (*e.g.*, Figure 1.5b). These stochastic variations to local textures constitute another more subtle form of multi-modality. Training I2I translation networks can suffer from a different version of mode collapse where the generator maps an input image only to a single possible output and ignores the multi-modal nature of such mapping. In this dissertation, we start by addressing both the training stability and style multi-modality challenges of I2I translation. Specifically, in Chapter 2, we propose a novel pre-training strategy to stabilize the training and reduce mode collapse, while also improving the performance of I2I networks in terms of diversity and output quality.

1.4 Applications of Neural Rendering for Novel View Synthesis (NVS)

Controllable and photo-realistic image synthesis opens up the door to improving and solving many real-world problems, such as image and video editing, visual effects, generating realistic simulation environments, data augmentation, and many more. Utilizing and extending Neural Rendering techniques to such applications is an ongoing research effort. In this dissertation, we

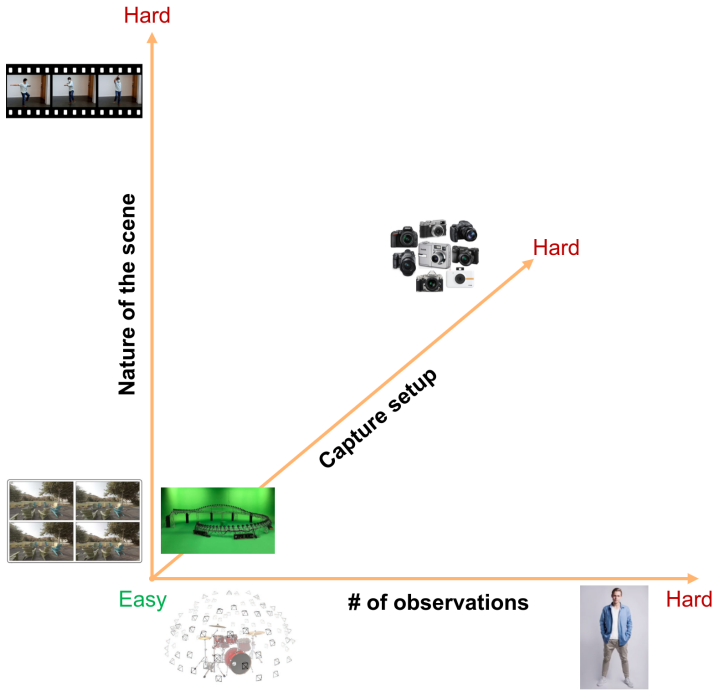


Figure 1.7: Novel View Synthesis tasks can be categorized along several axes, such as the number of observations of the scene (*e.g.*, many-/few-/single- shot), the nature of the scene (*e.g.*, static *v.* dynamic), and the capture setup (*e.g.*, camera array, single camera, random cameras).

investigate applying Neural Rendering techniques to the task of Novel View Synthesis.

Given a sparse set of observations of an object or a scene, Novel View Synthesis (NVS) aims to generate images of the scene from new unobserved views (Figure 1.6). However, the Novel View Synthesis problem has different variations, varying in difficulty along several axes (Figure 1.7).

One such axis is the capture setup used to record observations of the target object or scene. For example, we can have an expensive capture setup with an array of calibrated and synchronized cameras observing the scene from multiple views simultaneously. This multi-view capture setup provides very rich information that enables scene reconstruction and synthesizing new views by interpolating between different cameras. A more challenging setting is having a single calibrated camera which we move around the scene and capture images. In this setting, we

know the camera parameters which enables easy and consistent aggregation of observed views, by transforming observed images from image space into world coordinates via a simple matrix multiplication. An even more challenging setting is having the observations captured by a bunch of different uncalibrated cameras with unknown parameters. This requires solving a very large optimization problem to simultaneously estimate the parameters of each camera and align captured observations.

Another axis for classifying NVS tasks is the number of sampled images of the scene. For example we can have many observations that span different views of the scene. Or we can have few shots or even a single shot of a scene, where we are still required to generalize well to unobserved views. This requires developing a prior over different object categories so that we can imagine how unobserved views could look like.

A third axis for NVS is the nature of the scene itself. On one hand, we can have a static scene with static appearance, which is the simplest setting for NVS. Or we can have some static structure, but with dynamic appearance and with transients and occluders. For example, we can have a photo collection of a building or a tourist landmark, where images are taken at different times of days or under different lighting and weather conditions. In addition, images can contain transient objects and occluders like people or cars in front of the scene. And on the more challenging end of the spectrum, we can have a dynamic scene where different elements of the scene can vary between different captured observations. (*e.g.*, a human performer moving freely in front of a camera). This is very challenging since there is no direct multi-view consistency between different observations. Instead, we need to track and reason about moving parts of the scene so as to aggregate the observations into a consistent representation of the target scene. Figure 1.7 outlines the aforementioned axes for classifying different variations of Novel View Synthesis.

Table 1.2: Outline for the proposed Neural Rendering applications for Novel View Synthesis.

Novel View Synthesis Setting			Representation	Rendering	Chapter
# observations	**Capture setup**	**Scene nature**			
Many	Uncalib. cameras	Static + Dyn. appearance	Point cloud	Neural net	Chapter 3
Single/Few	Single uncalib. cam.	Dynamic	Facial landmarks + learned map	Neural net	Chapter 4
Many	Single calib. cam.	Static	Voxels + learned features	Vol. rendering	Chapter 5

Novel View Synthesis is integral to many modern applications such as augmented and virtual reality (AR/VR), telepresence, virtual tours, immersive interaction with photos, e-commerce and many more. In all of these applications, we would like to be able to move a virtual camera around and explore the scene from new angles. For example, in AR/VR, a user wearing a headset should be able to move their head and explore the environment around them. In telepresence, the goal is to provide an immersive 3D communication experience where visual content is rendered based on where the tracked eyes of a user are looking. In virtual tours, users want control a virtual camera to freely navigate through and explore a target scene. This ability to explore a scene from novel viewpoints requires two steps:

```
1. Recover some representation of the scene, which can be rotated
 and manipulated (ideally in 3D).
2. Render this representation from novel viewpoints to generate
 photo-realistic images.
```

Listing 1.1: Two-step general approach to Novel View Synthesis.

These two steps for generating photo-realistic novel views will be the general approach this dissertation explores for solving Novel View Synthesis under different settings of the three axes of Figure 1.7. Table 1.2 shows an outline for chapters of this dissertation that tackle NVS. In Chapter 3, we study NVS from in-the-wild Internet photos, where images exhibit extreme variation in appearance (*e.g.*, lighting, time of day, and weather conditions) and contain transient

occluders such as people, cars and different objects in front of a scene. Next, in Chapter 4, we study NVS in the single-shot/few-shot setting, where only a single or a handful of observations of a target subject is available, and the goal is to generate plausible images of this subject under novel head poses and facial expressions. Finally, in Chapter 5, we study NVS in the traditional setting, where we have a static scene with many captured observations, but we utilize a learned local representation to achieve high quality synthesis, following the success of *Implicit Neural Representations* [37, 38, 39, 40, 41].

Chapter 2: StEP: Style-based Encoder Pre-training for Multi-modal Image-to-Image Translation

Generative Adversarial Networks (GANs) [5] can model a given distribution of images. After a GAN is trained, it can generate new random samples from the target image distribution, but such setting lacks user control over the generated samples. To provide a form of user control, image-to-image (I2I) translation [7] conditions the synthesis process of GANs on a user provided image. I2I translation thus aims to translate images from an input domain to an output domain, such as user sketches to photo-realistic images or semantic maps to natural images. To tackle the multi-modal version of I2I translation, where input and output domains have a one-to-many relation, an extra latent input is provided to the generator to specify a particular output mode. Prior works [2, 42, 43] propose involved training objectives to learn a latent embedding, jointly with the generator, that models the distribution of possible outputs. Alternatively, in this chapter, we study a simple, yet powerful weakly-supervised pre-training strategy for multi-modal I2I translation. We first pre-train an encoder, using a proxy task, to encode the style of an image, such as color and texture, into a low-dimensional latent style vector. Then we train a generator to transform an input image along with a style-code to the output domain. Our generator achieves state-of-the-art results on 6 challenging benchmarks with a training objective that includes just a GAN loss and a reconstruction loss, which simplifies and speeds up the training significantly

compared to previous works. We show that the proposed style-based pre-training learns a more expressive latent space that achieves more faithful style capture and transfer. We further show that the learned style embedding is not dependent on the target domain and generalizes well across a wide range of domains, and finally we study the contribution of different loss terms to learning the task of multi-modal I2I translation.

2.1 Introduction

Image-to-image (I2I) translation is the task of transforming images from one domain to another (*e.g.*, semantic maps $\rightarrow$ scenes, sketches $\rightarrow$ photo-realistic images, ... etc). Many problems in computer vision and graphics can be cast as I2I translation, such as photo-realistic image synthesis [7, 44, 45], super-resolution [25], colorization [23, 24], and inpainting [22]. Therefore, I2I translation has recently received significant attention in the literature. One main challenge in I2I translation is the multi-modal nature for many such tasks – the relation between an input domain A and an output domain B is often times one-to-many, where a single input image $I_i^A \in A$ can be mapped to different output images from domain B. For example, a sketch of a shoe or a handbag can be mapped to corresponding objects with different colors or styles, or a semantic map of a scene can be mapped to many scenes with different appearance, lighting and/or weather conditions. Since I2I translation networks typically learn one-to-one mappings due to their deterministic nature, an extra input is required to specify an output mode to which an input image will be translated. Simply injecting extra random noise as input proved to be ineffective as shown in [2, 7], where the generator network just learns to ignore the extra noise and collapses to a single or few modes (which is one form of the mode collapse problem). To overcome this problem, Zhu *et al.* [2] proposed *BicycleGAN*, which trains an encoder network E, jointly with the I2I

translation network, to encode the distribution of different possible outputs into a latent vector z, and then learns a deterministic mapping $G : (A, z) \rightarrow B$. So, depending on the latent vector z, a single input $I_i^A \in A$ can be mapped to multiple outputs in B. While BicycleGAN requires paired training data, several works [42, 43] extended it to the unsupervised case, where images in domains A and B are not in correspondence ('unpaired'). One main component of unpaired I2I translation is a cross-cycle consistency constraint, where the network generates an intermediate output by swapping the styles of a pair of images, then swaps the style between the intermediate output again to reconstruct the original images. This enforces that the latent vector z preserves the encoded style information when translated from an image i to another image j and back to image i again. This constraint can also be applied to paired training data, where it encourages style/attribute transfer between images. However, training BicycleGAN [2] or its unsupervised counterparts [42, 43] is not trivial. For example, BicycleGAN combines the objectives of both conditional Variational Auto-Encoders (cVAEs) [46] and a conditional version of Latent Regressor GANs (cLR-GANs) [47, 48] to train their network. The training objective of [42, 43] is even more involved to handle the unsupervised setup.

In this chapter, we propose a novel pre-training strategy to learn an expressive latent space for the task of multi-modal I2I translation. While end-to-end training of the encoder network E with the I2I translation network poses a convenience, we show that it can be advantageous, both in terms of training stability and output quality for downstream tasks, to break down the training into proxy tasks. In specific, we show both quantitatively an qualitatively that the proposed pre-training yields the following advantages:

- It learns a more powerful and expressive latent space. Specifically, we show that: (1) Our

pre-trained latent space captures rare/extreme styles that are not well represented in the training set, while BicycleGAN-based baselines fail to do so and instead tend to simplify such styles/appearances to the nearest common style in the train set. (2) Pre-training yields more faithful style capture and transfer. (3) Finally, the better expressiveness of the pre-trained latent space leads to more complex style interpolations compared to BicycleGAN-based baselines.

- The learned style embedding is not dependent on the target dataset, and generalizes well across multiple domains, which is especially useful for the case of having limited training data.

- Style pre-training simplifies the training objective by requiring fewer losses, which also speeds up the training.

- Our approach improves the training stability and the overall output quality and diversity.

We note that our proposed style pre-training is weakly-supervised and doesn't require any manual labeling. Instead it relies on a pre-trained VGG network [49] to provide training supervision. Our work is inspired by the standard training paradigm in visual recognition of first pre-training on a proxy task, either large supervised datasets (*e.g.*, ImageNet) [50, 51, 52] or unsupervised tasks (*e.g.*, [53, 54]), and then fine-tuning (transfer learning) on the desired task. Similarly, we propose to pretrain the encoder using a weakly-supervised proxy task that encourages capturing style into a latent space. Our goal is to highlight the benefits of encoder pre-training and demonstrate its effectiveness for multi-modal image synthesis. In particular, we make the following contributions:

- We explore style pre-training and its effectiveness for the task of multi-modal I2I translation, which simplifies and speeds up the training compared to competing approaches.
- We show that the pre-trained latent embedding is not dependent on the target domain and generalizes well to other domains (transfer learning).
- We provide a study of the importance of different losses and regularization terms for multi-modal I2I translation networks.
- We achieve state-of-the art results on several benchmarks in terms of style capture and transfer, and diversity of results.

2.2 Related work

Deep generative models. There has been incredible progress in the field of image synthesis using deep neural networks. In its unconditional setting, a decoder network learns to map random values drawn from a prior distribution (typically Gaussian) to output images. Variational Auto-Encoders (VAEs) [55] assume a bijection mapping between output images and some latent distribution and learn to map the latent distribution to a unit Gaussian using the reparameterization trick. Alternatively, Generative Adversarial Networks (GANs) [5] directly map random values sampled from a unit Gaussian to images, while using a discriminator network to enforce that the distribution of generated images resembles that of real images. Recent works proposed improvements to stabilize the training [35, 36, 56, 57] and improve the quality and diversity of the output [11, 13]. Other works combine both VAEs and GANs into a hybrid VAE-GAN model [58, 59].

Conditional image synthesis. Instead of generating images from input noise, the generator can

be augmented with side information in the form of extra conditional inputs. For example, Sohn *et al.* [46] extended VAEs to their conditional setup (cVAEs). Also, GANs can be conditioned on different information, like class labels [60, 61, 62], language description [63, 64], or an image from another domain [7, 44]. The latter is called image-to-image translation.

Image-to-image (I2I) translation. I2I translation is the task of transforming an image from one domain, such as a sketch, into a another domain, such as photo-realistic images. While there are regression-based approaches to this problem [44, 65], significant successes in this field are based on GANs and the influential work of pix2pix [7]. Following the success of pix2pix [7], I2I translation has since been utilized for a large number of tasks, like inpainting [22], colorization [23, 24], super-resolution [25], rendering [27, 28, 29], and many more [30, 31, 32]. There has also been works to extend this task to the unsupervised setting [65, 66, 67, 68, 69, 70], to multiple domains [71], and to videos [72, 73].

Multi-modal I2I translation. Image translation networks are typically deterministic function approximators that learn a one-to-one mapping between inputs and outputs. To extend I2I translation to the case of diverse multi-modal outputs, Zhu *et al.* [2] proposed the BicycleGAN framework that learns a latent distribution that encodes the variability in the output domain and conditions the generator on this extra latent vector for multi-modal image synthesis. Wang *et al.* [45, 73] learn instance-wise latent features for different objects in a target image, which are clustered after training to find f fixed modes for different semantic classes. At test time, they sample one of the feature clusters for each object to achieve multi-modal synthesis. Other works extended the multi-modal I2I framework to the unpaired setting, where images from the input and output domains are not in correspondence [42, 43, 74], by augmenting BicycleGAN with different forms of a cross-cycle consistency constraint between two unpaired images. In our work,

we focus on the supervised setting of multi-modal I2I translation. We propose a self-supervised pre-training strategy to learn a latent distribution that encodes variability in the output domain. The learned distribution can be easily adapted to new unseen datasets with simple fine-tuning, instead of training from a random initialization.

2.3 Approach

Current multi-modal image translation networks require an extra input z that allows for modelling the one-to-many relation between an input domain A and an output domain B as a one-to-one relation from a pair of inputs $(A, z) \rightarrow B$. In previous approaches, there has been a trade-off between simplicity and effectiveness for providing the input z. On one hand, providing random noise as the extra input z maintains a simple training objective (same as in pix2pix [7]). However, [2, 7] showed that the generator has little incentive to utilize the input vector z since it only encodes random information, and therefore the generator ends up ignoring z and collapsing to one or few modes. On the other hand, *BicycleGAN* [2] combines the objectives of both conditional Variational Auto-Encoder GANs (cVAE-GAN) and conditional Latent Regressor GANs (cLR-GAN) to learn a latent embedding z simultaneously with the generator G. Their training enforces two cycle consistencies: $B \rightarrow z \rightarrow \hat{B}$ and $z \rightarrow \tilde{B} \rightarrow \hat{z}$. This proved to be very effective, but the training objective is more involved, which makes the training slower. Also, since the latent embedding is being trained simultaneously with the the generator, hyper-parameter tuning becomes more critical and sensitive.

We aim to combine the best of both worlds: an effective training of a latent embedding that models the distribution of possible outputs, while retaining a simple training objective. This

would allow for faster and more efficient training, as well as less sensitivity to hyper-parameters. We observe that the variability in many target domains can be represented by the style diversity of images in the target domain B, where the style is defined in terms of the Gram matrices used in the Neural Style Transfer literature [75]. Then, we learn an embedding by separately training an encoder network E on an auxiliary task to optimize for $z = E(I^B)$ capturing the style of an image I^B. Visualizing the pre-trained latent space shows the that pre-trained encoder models different modes of the output distribution (such as different colors, lighting and weather conditions) as clusters of images with similar styles as shown in § 2.4.7. Then, to synthesize an image $\hat{B} = G(I^A, z)$, the input latent can be used to clearly distinguish the style cluster to which the output belongs. This makes for an effective and more stable training of the generator G, since G is just required to discover the correlation between output images and their corresponding style embedding z. Moreover, experimental evaluation shows that style-based pre-training yields better results in terms of more faithful style capture and transfer, as well as better output quality and diversity.

To incorporate this into BicycleGAN [2], we replace the simultaneous training of the encoder E and the generator G with a staged training (Figure 2.1) as follows:

- **Stage 1**: Pre-train E on a proxy task that optimizes an embedding of images in the output domain B into a low-dimensional style latent space, such that images with similar styles lie closely in that space (*i.e.* clustered).

- **Stage 2**: Train the generator network G while fixing the encoder E, so that G learns to associate the style of output images to their deterministic style embedding $z = E(I^B)$.

- **Stage 3**: Fine-tune both the E and G networks together, allowing for the style embedding

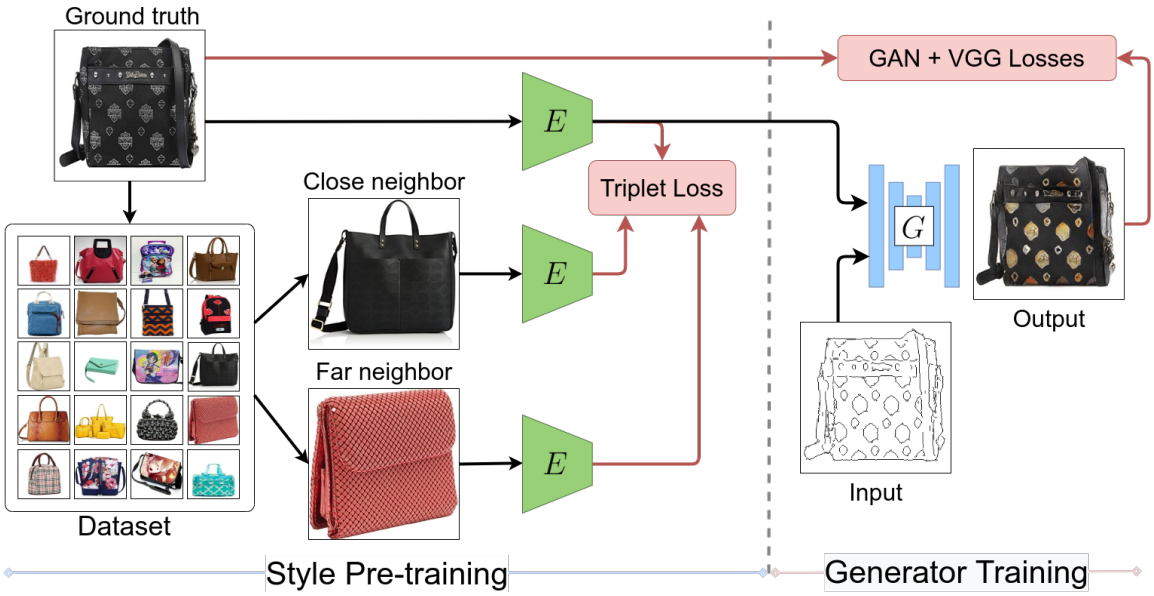


Figure 2.1: Overview of our training pipeline. Stage 1: pretraining the style encoder E using a triplet loss. Stages 2, 3: training the generator G, and finetuning both G, E together using *GAN* and reconstruction losses.

to be further adapted to best suit the image synthesis task for the target domain.

Next, we explain how to pre-train the style encoder network E in § 2.3.1, and how to train the generator G using the pre-learned embeddings (§ 2.3.2). Finally, we demonstrate the generalization of pre-training the style encoder E in § 2.3.3.

2.3.1 Weakly-supervised pre-training of the style encoder E

The goal of pre-training the encoder network E is to learn a deterministic mapping from the style of a target image $I_i^B \in B$ to a latent style code $z_i = E(I_i^B)$. Ideally, images with similar styles should be close in the style embedding space, while images with different styles should be far from each other. To supervise training such embedding, we utilize the style loss [75] as a distance metric to measure the style similarity between any two given images. The style encoder network E is then trained using a triplet loss [76], where the input is a triplet of images

(I_a, I_p, I_n), where (I_a, I_p) have a similar style, while (I_a, I_n) have different styles, as measured by the style loss metric. The training objective for E is given by:

$$\mathcal{L}^{\text{tri}}(I_a, I_{\text{p}}, I_{\text{n}}) = \max\left(\left[\|z_a - z_p\|^2 - \|z_a - z_n\|^2 + \alpha\right], 0\right) + \lambda \mathcal{L}^{\text{reg}}\left(z_a, z_p, z_n\right) \tag{2.1}$$

where α is a separation margin, λ is a relative weighting parameter between the main triplet objective and an optional regularization term $\mathcal{L}^{\text{reg}}(\cdot)$ which is an $L2$ regularization to encourage learning a compact latent space.

Triplet selection. To generate triplets for pre-training the encoder E, we compute the set of k_c closest and k_f furthest neighbors for each anchor image I_a as measured by the style loss [75]. Then, for each anchor image I_a, we randomly sample a positive image I_p and a negative image I_n from the set of closest and furthest neighbors respectively. We found that, for large datasets, it is sufficient to generate triplets for a subset of the training images. One challenge is the set of images with an outlier style. Such images will be furthest neighbors to most images, and can mislead the training by just projecting outlier images to separate clusters. To deal with this, we sample the negative style image I_n from a larger set of furthest neighbors; while the positive image I_p is sampled from a small set of closest neighbors so that it would have reasonable style similarity to the anchor image.

2.3.2 Generator training

After pre-training the style encoder E (stage 1), we have established a mapping from images in the output domain, $I^B \in B$, to their style-embedding $z = E(I^B)$. Feeding the style embedding as input to the generator during training, the generator has good incentive to associate

the style of output images to their corresponding style embedding instead of learning to hallucinate the style. It's important to retain the deterministic correspondence between images and their style codes to facilitate the job of the generator to discover this correlation. This is why, during stage 2, we keep the weights of the style encoder E fixed. The forward pass reconstructs a training image I_i^B as $\hat{I}_i^B = G(I_i^A, z_i)$, where $z_i = E(I_i^B)$. The training objective is similar to that of pix2pix [7],

$$\mathcal{L}^{\text{img}}(I_i^B, \hat{I}_i^B) = \mathcal{L}_{\text{cGAN}}(I_i^B, \hat{I}_i^B) + \lambda_{\text{rec}}\mathcal{L}_{\text{rec}}(I_i^B, \hat{I}_i^B) \tag{2.2}$$

where we use the Least Square GAN loss (LSGAN) [57] for the $\mathcal{L}_{\text{cGAN}}$ term, and a VGG-based perceptual loss [49] for the reconstruction term $\mathcal{L}_{\text{rec}}$. Once the generator has learned to associate the output style with the input style embedding, stage 3 fine-tunes both the generator G and the style encoder E together using the same objective (2.2).

Style sampling. To perform multimodal synthesis on a given input at test time, we can capture the latent vector z from any existing image and transfer the style to the generated image. However, if we wish to sample styles directly from the latent distribution, one option is to enforce a prior on the latent distribution. For example, we found it effective to add an $L2$ regularization on the the latent vectors to enforce zero-mean embeddings and limit the variance of the latent space. We then compute an empirical standard deviation for sampling. Another alternative to enable sampling is to train a mapper network $\mathcal{M}$ to map the unit Gaussian to the latent distribution. This can be done as a post-processing step after the style encoder has been trained and fine-tuned. Specifically, we propose to train a mapper network $\mathcal{M}$ using the nearest-neighbor based Implicit

Maximum Likelihood Estimation (IMLE) training [77, 78]. The training objective is given by:

$$\mathcal{M} = \arg\min_{\tilde{\mathcal{M}}} \sum_i \|z_i - \tilde{\mathcal{M}}(e_i)\|_2^2, \quad e_i = \arg\min_{r_j} \|z_i - \mathcal{M}(r_j)\|_2^2 \tag{2.3}$$

where $\{r_j\}$ is a set of random samples from the unit Gaussian prior, and for each latent code z_i, we select e_i that maps to the nearest neighbor $\mathcal{M}(e_i)$ to z_i.

2.3.3 Generalizing the pre-training stage

The use of Gram matrices for Neural Style Transfer proved to be very effective and it reliably captures the style of arbitrary input style images. This implies that Gram matrices can reliably encode the styles from a wide range of domains, and they are not specific to a certain domain. Therefore, we hypothesize that encoder pre-training using a style-based triplet loss would learn a generic style embedding that can generalize across multiple domains and be effective for multi-modal I2I translation. This would allow for performing the pre-training stage only once using auxiliary training data. The fine-tuning stage eventually tweaks the embedding to better suit the specific target domain B. We validate our hypothesis experimentally in § 2.4, and show that pre-training the style encoder on datasets other than the target domain B doesn't degrade the performance. It can even improve the performance if the target dataset is small, in which case pre-training on an auxiliary dataset helps with the generalization of the overall model.

2.3.4 Implementation details

The generator network G has a symmetric encoder-decoder architecture based on [45], with extra skip connections by concatenating feature maps of the encoder and decoder. We

Figure 2.2: Qualitative comparison with baselines. Ours better matches the ground truth (GT) style.

use a multiscale-patchGAN discriminator [45] with 3 scales and employ a LSGAN [57] loss. The mapper network $\mathcal{M}$ is a multi-layer perceptron (MLP) with three 128-dimensional hidden layers and a *tanh* activation function. For the reconstruction loss, we use the perceptual loss [49] evaluated at $\texttt{conv}_{i,2}$ for $i \in [1, 5]$ of VGG [79] with linear weights of $w_i = 1/2^{6-i}$ for $i \in [1, 5]$. The architecture of the style encoder E is adopted from [43], and we use a latent style vector $z \in \mathbb{R}^8$. Our optimizers setup is similar to that in [2]. We use three *Adam* optimizers: one for the generator G and encoder E, another for the discriminator D, and another optimizer for the generator G alone with $\beta_1 = 0, \beta_2 = 0.99$ for the three optimizers, and learning rates of $0.001, 0.001$ and 0.0001 respectively. We use a separate *Adam* optimizer for the mapper network $\mathcal{M}$ with $\beta_1 = 0.5, \beta_2 = 0.99$, and a learning rate of 0.01 with a decay rate of 0.7 applied every 50 steps. Relative weights for the loss terms are $\lambda_{\text{cGAN}} = 1$, $\lambda_{\text{rec}} = 0.02$ and $\lambda_{L2} = 0.01$ for the GAN loss, reconstruction loss, and $L2$ latent vector regularization respectively. When sampling triplets for any anchor image I_c, we use $k_c = 5, k_f = 13$ for the size of the set of close and far neighbors respectively.

Table 2.1: Validation set reconstruction quality, as measured by *PSNR* (higher is better) and *LPIPS* [1] (lower is better), for various datasets. We compare between retraining BicycleGAN [2] authors' released code (Bicycle v0), our implementation of the two baselines (BicycleGAN v1 and MUNIT-p) described in Section 2.4, and our approach both before fine-tuning (ours - stage 2), and after fine-tuning (ours - stage 3).

	edge2shandbags		edges2shoes		labels2facades		night2day		maps		space needle	
	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓
Bicycle v0	17.08	0.255	20.24	0.177	12.64	0.431	13.25	0.520	14.32	0.396	–	–
Bicycle v1	18.52	0.198	21.84	0.124	13.08	0.378	13.88	0.491	14.67	0.359	19.72	0.233
MUNIT-p	**19.23**	0.192	22.51	0.132	13.36	0.375	14.48	0.480	**16.17**	0.407	19.84	0.238
ours - stage 2	18.01	0.209	21.40	0.140	**13.44**	0.383	14.34	**0.476**	15.08	0.392	**21.39**	**0.227**
ours - stage 3	18.91	**0.177**	**22.68**	**0.117**	**13.44**	**0.370**	**15.05**	**0.452**	15.15	**0.349**	**22.11**	**0.187**

2.4 Experimental evaluation

Datasets. We evaluate our approach on five standard I2I translation benchmarks used in [2, 7]; Architectural labels → photo, aerial → map, edges → shoes/handbags and night → day. In addition, we use the Space Needle timelapse dataset [80], which consists of 2068 paired images with a 8280×1080 resolution, where the input domain includes images with temporally smoothed appearance, and the output domain contains real images spanning different lighting and weather conditions.

Baselines. While we report numbers for retrained models using the official code released with BicycleGAN (BicycleGAN v0) for completeness, we mainly compare to two stronger baselines:

- **BicycleGAN v1**: we implement BicycleGAN using the same network architecture as used in our approach to have a fair comparison.

- **MUNIT-p**: We train MUNIT [42] in a *paired* setup by applying its cross-cycle consistency constraint as follows: the input is a pair of training examples $(I_1^A, I_1^B), (I_2^A, I_2^B)$ for which we obtain their respective style embeddings $z_1 = E(I_1^B), z_2 = E(I_2^B)$. We then apply a

Figure 2.3: Style transfer for different datasets. For each dataset, we show output for applying different styles to the same input image.

2-step cyclic reconstruction of I_1^B, I_2^B; in the first step, we generate both images with a swapped style $u = G(I_1^A, z_2), v = G(I_2^A, z_1)$. In the second step, we re-capture the latent style vectors $\hat{z}_2 = E(u), \hat{z}_1 = E(v)$ and generate the original images I_1^B, I_2^B by swapping the style again: $\hat{I}_1^B = G(I_1^A, \hat{z}_1)$, $\hat{I}_2^B = G(I_2^A, \hat{z}_2)$. We add a cyclic reconstruction loss for $\hat{I}_1^B, \hat{I}_2^B$.

2.4.1 Image reconstruction

We report the reconstruction quality of validation set images, using both *PSNR* and *AlexNet*-based *LPIPS* [1] metrics, in Table 2.1. Note that our results with the pre-trained embeddings without fine-tuning (stage 2) are on-par-with the baselines. This verifies the validity of our approach and that style-based encoder pre-training successfully learns to distinguish different style modes in the output domain, which proves effective for training multi-modal I2I networks. Fine-tuning (stage 3) further improves our results compared to the baselines. Figure 2.2 shows qualitatively how our approach reconstructs the target style more faithfully. Our approach not only matches

Figure 2.4: Style sampling for different datasets using our approach v3. We sample either from $N(\mu, \sigma)$, where μ, σ are computed from the train set (middle), or using the mapper network $\mathcal{M}$ (right).

the ground truth colors better, but also texture (*e.g.*, left column, first and third rows). While the baselines rely on VAEs to provide the style sampling property, we observe that, for a low dimensional latent space, the noise robustness added by VAEs comes at the expense of the expressiveness of the latent space. This explains why our approach, which does not use VAEs, achieves more faithful style capture and reconstruction. We verified this by removing the VAE component from the baselines, resulting in improving their performance as shown in § 2.4.5.

2.4.2 Style transfer and sampling

Figure 2.3 shows style transfer to images from the validation set of different datasets. Note how the style transfer copies the weather conditions in the Space Needle and Night2day

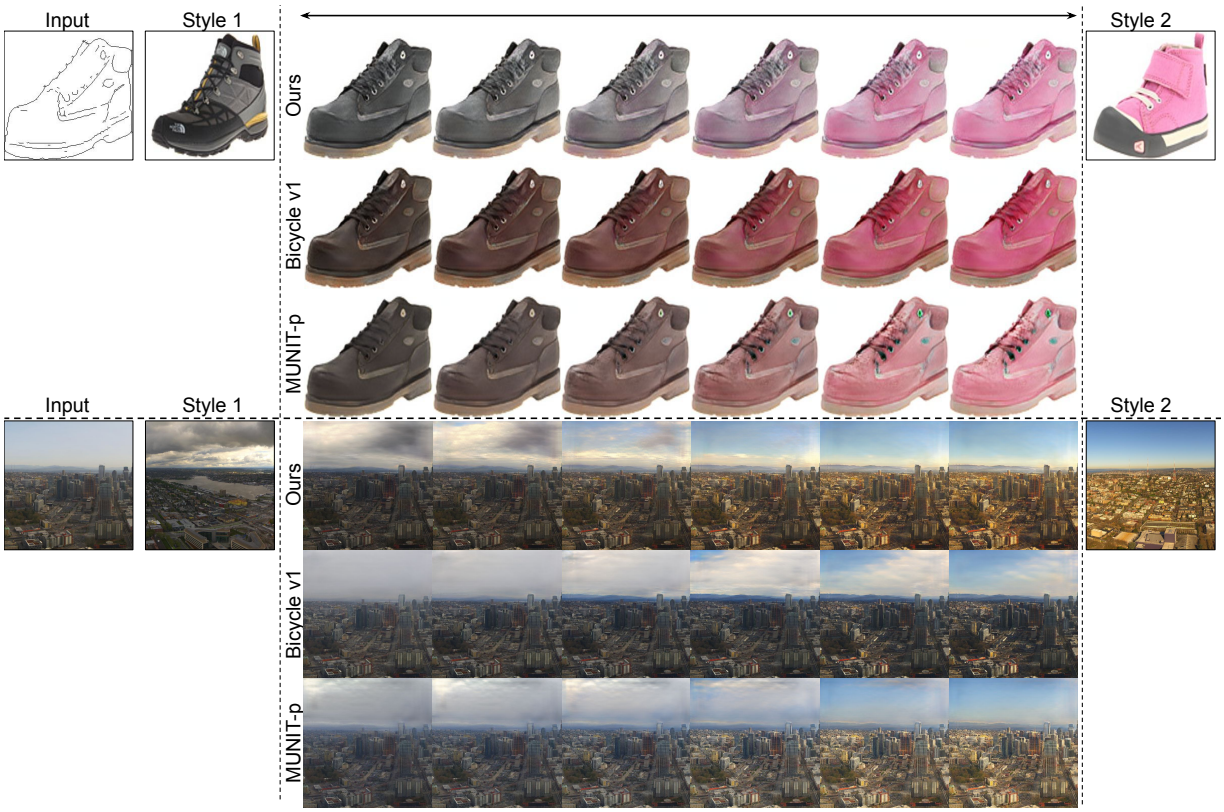


Figure 2.5: Style interpolation. Left column is the input to the generator G, second and last columns are input style images to the style encoder, and the middle images are linear interpolation in the embedding space (figure better seen in zoom).

datasets. For example, in the Space Needle dataset, we show sunset, sunny, foggy and cloudy weather. Also, the Night2day examples exhibit variation in lighting conditions including transferring whether the surface is sunlit or not, as well as different cloud patterns and clear skies. We can also sample random styles directly from the the latent distribution as described in § 2.3.2. Figure 2.4 shows results for both adhoc sampling from the assumed $N(\mu, \sigma)$ empirical distribution, as well as formally sampling from a unit Gaussian using the mapper network $\mathcal{M}$. Note that the diversity of sampled styles does not stem from simple color changes; for example, sampled styles for the Space Needle dataset show clear weather changes, such as cloudy vs sunny, change in cloud patterns and even sampling foggy weather which was present in some images in the training set. In the Maps dataset, the existence and/or density of bushes clearly varies between

different sampled styles. Also, in the Edges2handbags dataset, the texture of the bag varies between smooth and rough leather (better seen in zoom). While the results of both sampling methods look good, we note that the assumption for adhoc sampling is not explicitly enforced, and thus could sample bad style codes outside the distribution (see § 2.4.13 for examples).

2.4.3 Style interpolation

Figure 2.5 shows style interpolation by linearly interpolating between two latent vectors. Besides matching the start and target styles better than the baselines, our approach shows more complex interpolation. For example, note the smooth change in lighting and cloud patterns when going from cloudy to sunny in the Space Needle dataset.

Table 2.2: Generalization of a pretrained style encoder E. We report validation set reconstruction metrics for the edges2handbags and night2day datasets when pre-training with different datasets. Stages 2, 3 show results before/after fine-tuning E respectively.

Dataset	pre-train dataset	Stage 2		Stage 3	
		PSNR ↑	LPIPS ↓	PSNR ↑	LPIPS ↓
edges2handbags	edges2handbags	18.01	0.209	18.91	0.177
	edges2shoes	17.89	0.215	18.96	0.176
	space_needle	17.86	0.221	19.02	0.175
night2day	night2day	13.75	0.489	15.15	0.454
	space_needle	14.34	0.476	15.05	0.452
	edges2handbags	13.91	0.492	15.03	0.461

2.4.4 Pre-training generalization

Since the notion of style, as defined in the Neural Style Transfer literature, is universal and not specific to a certain domain, we hypothesized that style-based encoder pre-training would learn a generic style embedding that can generalize across multiple domains and be effective for

multi-modal I2I translation. Here, we experimentally verify our hypothesis in Table 2.2. For a target dataset, we train the generator G three times, each with a different pre-training of the style encoder E: (1) same dataset pre-training: pre-train E using the output domain B of the target dataset. (2) similar-domain pre-training: pre-train on a different dataset, but whose output domain bears resemblance to the output domain of the target dataset (*e.g.*, edges2shoes and edges2handbags, or day images from night2day and the Space Needle timelapse dataset). (3) different-domain pre-training: pre-train on a different dataset whose output domain has different styles from that of the target dataset (*e.g.*, edges2handbags and the Space Needle timelapse datasets, or night2day and edges2handbags datasets). Table 2.2 shows that without fine-tuning (stage 2), the edges2handbags dataset shows a slight performance degradation when going from pre-training on the same dataset, to pre-training on a similar-domain dataset, and finally pre-training on a different-domain dataset. On the other hand, the night2day dataset has only $\sim$100 unique scenes for training. So, pre-training on another dataset such as Space Needle generalizes better to new scenes in the validation set, since it helps avoid overfitting the small number of unique scenes in the training set. After fine-tuning, performance differences further reduce to be insignificant. We also investigate the generalization of the proposed encoder pre-training to the case of using non-style distance metrics in § 2.4.12.

2.4.5 Ablative study

We investigate the role of different loss terms as we transition from the loss setup of the baselines to that of our training approach. We first remove the variational part in both the BicycleGAN v1 and MUNIT-p baselines resulting in Bicycle v2 and MUNIT-p v2. We further remove

Table 2.3: Ablation study of the effect of different components and loss terms using the edges2handbags dataset. We study direct and cyclic reconstructions on ground truth images (dir_recon, cyc_recon), discriminator loss on direct reconstructions and on generated images with a randomly sampled style (D_dir, D_rand_z), latent reconstruction (z_recon), $L2$ and KL regularization on the latent vector z (z_L2, z_KL), and finally the use of VAE vs. just an auto-encoder.

Approach	Loss setup								PSNR↑	IS↑	LPIPS ↓
	dir_recon	cyc_recon	D_dir	D_rand_z	z_recon	z_L2	z_KL	VAE			
Bicycle v1	✓	–	✓	✓	✓	–	✓	✓	18.28 ± 0.30	2.31 ± 0.05	0.201 ± 0.003
MUNIT-p	✓	✓	✓	✓	✓	–	✓	✓	18.96 ± 0.30	2.45 ± 0.07	0.192 ± 0.002
Bicycle v2	✓	–	✓	✓	✓	–	✓	–	19.02 ± 0.10	2.36 ± 0.12	0.175 ± 0.001
MUNIT-p v2	✓	✓	✓	✓	✓	–	✓	–	19.34 ± 0.07	2.44 ± 0.06	0.176 ± 0.002
Bicycle v3	✓	–	✓	✓	✓	✓	–	–	19.21 ± 0.06	2.34 ± 0.08	0.177 ± 0.002
MUNIT-p v3	✓	✓	✓	✓	✓	✓	–	–	19.24 ± 0.09	2.33 ± 0.04	0.180 ± 0.001
Ours v1	✓	✓	✓	✓	✓	✓	–	–	18.97 ± 0.13	2.41 ± 0.07	0.189 ± 0.004
Ours v2	✓	–	✓	✓	✓	✓	–	–	18.94 ± 0.10	2.43 ± 0.03	0.183 ± 0.002
Ours v3	✓	–	✓	–	–	✓	–	–	18.94 ± 0.05	2.42 ± 0.03	0.176 ± 0.001
Ours v4	✓	–	✓	–	–	–	–	–	18.94 ± 0.02	2.46 ± 0.03	0.177 ± 0.001

the Gaussian prior and replace the KL loss with an L2 regularization in Bicycle v3, MUNIT-p v3. To maintain random latent vector sampling during training without a prior, we sample a random training image, and use its style code. We define different versions of our approach (v1, v2, v3, and v4) based on different loss setup during training as follows: we start with 'Ours v1', which has the same loss setup as MUNIT-p v3, except that it uses pre-trained embeddings as described in § 2.3.1. We then remove cyclic reconstruction, random z sampling, and L2 regularization terms resulting in 'Ours v2', 'v3', and 'v4' respectively. We run each setup on the edges2handbags dataset. In order to draw more reliable conclusions, we repeat each experiment 3 times and report the mean and standard deviation in Table 2.3. We notice that removing the variational part in VAEs is enough to achieve good results. While VAEs in general are robust to noise in the input latent, we observe that this comes at the expense of the expressiveness of the latent space (*e.g.*, less faithful style capture and transfer), especially for low dimensional latents. We also observe that our approach generally performs better with less constraints (loss terms).

For example, "Ours v1, v2" have lower results than their "Bicycle v3", "MUNIT-p v3" counterparts. This shows that the main benefit of pre-trained embeddings is when the network is less constrained.

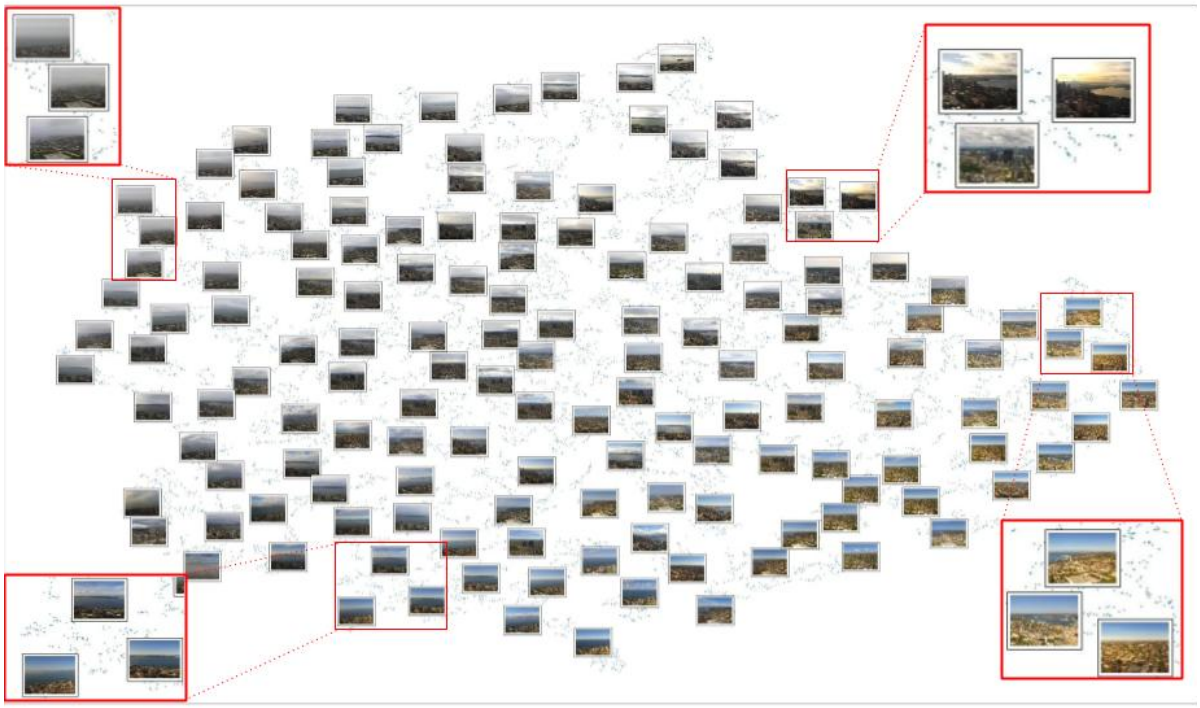
Table 2.4: Diversity score is the average LPIPS distance [1]. User preference score is the percentage a method is preferred over Ours v4, on the edges2shoes dataset.

Approach	LPIPS ↑ (transfer)	LPIPS ↑ (sampling)	User preference ↑
Bicycle v1	0.102	0.119	30.0%
MUNIT-p	0.138	0.132	37.7%
Ours v1	0.153	0.148	46.5%
Ours v2	0.171	0.140	41.1%
Ours v3	0.149	0.165	50.4%
Ours v4	0.154	0.132	50%

2.4.6 Diversity and user study

We evaluate diversity by computing the average LPIPS distance over 1600 output images. In this setting, higher LPIPS translates to higher diversity. We measure diversity on two setups: we sample 100 validation images, and (1) apply style transfer from 16 randomly sampled images, or (2) we sample 16 random codes using the mapper network $\mathcal{M}$ to obtain 1600 outputs. We also measure the realism and faithfulness of style transfer through a user study, where 30 participants are shown an input shoe sketch, an input style image and two style transfer outputs. They are given unlimited time to choose which output looks more realistic, and if both are realistic, then which transfers the style more faithfully. We fix 'Ours v4' approach as anchor and compare other methods to it. Table 2.4 shows that the baselines achieve lower diversity and user preference compared to our approach, specially in the style transfer setup. Different variations of our method, except for 'Ours v2' yield similar diversity and user preference scores. We observe that 'Ours

(a) Pre-trained embedding for the edges2shoes dataset.



(b) Pre-trained embedding for the Space Needle timelapse dataset.

Figure 2.6: t-SNE plots for the latent style space learned by the pre-trained style encoder E for two different datasets (figure best seen in zoom).

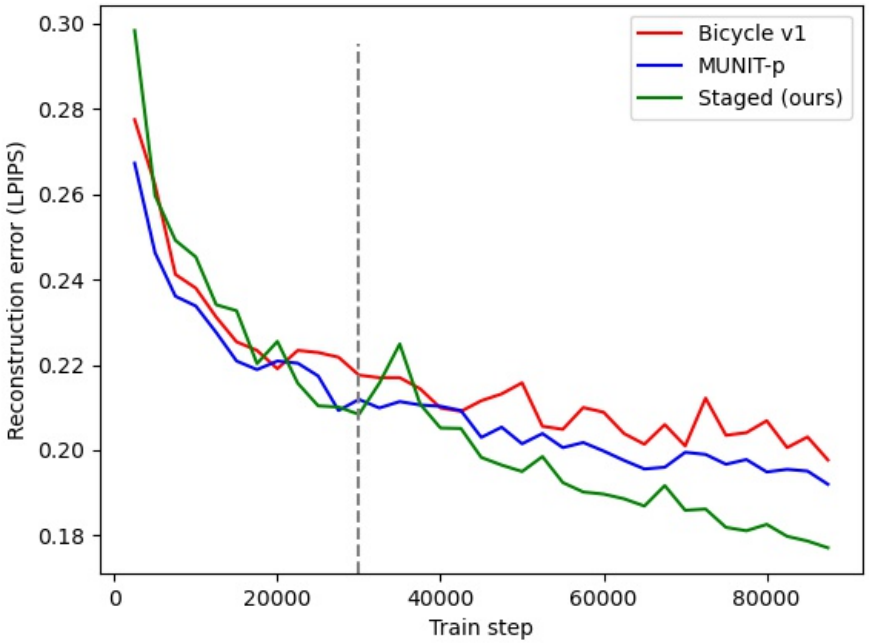


Figure 2.7: Convergence comparison between the proposed staged training (ours - v3) and the BicycleGAN-based baselines measured by the reconstruction error (LPIPS) of the validation set of the edges2handbags dataset. Dotted line shows the transition between stages 2 and 3 of our training (*i.e.*, switching from a fixed E to fine-tuning both G and E together).

v2' shows artifacts in some outputs, leading to higher diversity but lower user preference. Our diversity results for the style sampling setup have some variation and are sensitive the mapper network training, but are still either on-par or better than the baselines.

2.4.7 Visualizing pre-trained embeddings

Figure 2.6 visualizes the pre-trained latent space learned by the style encoder E. The visualization shows meaningful clusters of similar styles (colors for the edges2shoes dataset and weather conditions for the Space Needle timelapse dataset). We also visualize the latent space after fine-tuning as well as the latent space learned by BicycleGAN in § 2.4.11. However, the t-SNE plot visualization does not reveal significant differences.

2.4.8 Convergence analysis

Figure 2.7 compares the convergence of our staged training compared to the BicycleGAN baselines. The dotted line in the graph marks the transition between stages 2 and 3 of our training (*i.e.*, switching from a fixed pre-trained encoder E to fine-tuning both G and E together). We measure the reconstruction error (LPIPS) of the validation set of the edges2handbags dataset as the training progresses. Results show that with a fixed pre-trained encoder, our staged training starts with higher error than the baselines, but quickly drops to show similar performance as the baselines, and even beats the baselines before switching to stage 3 (marked by a dotted line). When starting to fine-tune the encoder E, we get a spike in the reconstruction error as the network adapts to the shift in the pre-trained embeddings, but then the performance of the staged training steadily widens the performance gap with the baselines. This shows the importance of the fine-tuning stage to tweak the pre-trained embeddings to better serve the image synthesis task.

Table 2.5: Training time (in seconds) per 1000 images for the baselines, as well as different versions of our approach (defined in Table 2.3).

Approach	Batch size	time/kimg↓ (sec)	Max batch size	time/kimg↓ (sec)
Bicycle v1	8	93.11	12	85.36
MUNIT-p	8	155.72	8	155.72
Ours v1	8	145.50	8	145.50
Ours v2	8	98.55	12	93.04
Ours v3	8	64.92	16	53.80
Ours v4	8	63.96	16	51.23

2.4.9 Training time

Simplifying the training objective allows for faster training time, as well as a larger batch size due to lower memory usage. Table 2.5 shows the processing time per 1000 training images for the baselines as well as different variations of our approach as defined in Table 2.3.

Table 2.6: Inception score comparison (higher is better) for different datasets.

	handbags	shoes	facades	night2day	maps	space needle
Bicycle v1	2.13	2.83	1.41	1.65	3.26	1.82
MUNIT-p	2.07	2.64	1.45	1.74	**3.57**	1.77
ours - stage 2	**2.22**	**2.75**	**1.61**	1.76	3.32	**1.90**
ours - stage 3	2.15	2.85	1.56	**1.84**	3.28	1.89

2.4.10 More quantitative comparison

We report the Inception Score (IS) computed over the validation set of various datasets in Table 2.6. Surprisingly, results after fine-tuning ("ours - stage 3") are slightly worse than those before fine-tuning ("ours - stage 2"), but both are still better than the baselines except for the 'Maps' dataset. We also note the Inception Score is not very suited to image-to-image translation tasks, since it prefers output diversity with respect to ImageNet classes, not the within-class diversity as in the I2I translation case.

2.4.11 Latent space comparison

Figure 2.8 visualizes the latent space learned by the style encoder E after pre-training and before fine-tuning (2.8a), after fine-tuning (2.8b), and the latent space learned by BicycleGAN [2]

(a) Our approach: after style pre-training.

(b) Our approach: after fine-tuning.

(c) BicycleGAN v1 baseline.

Figure 2.8: t-SNE plots for the latent style space learned by the style encoder E (a) after style pre-training, (b) after fine-tuning, and (c) using the BicycleGAN v1 baseline.

Figure 2.9: t-SNE plot for the pre-trained latent space learned for facial expressions on a subset of the KDEF dataset. Images with the same emotion or facial expression are correctly clustered.

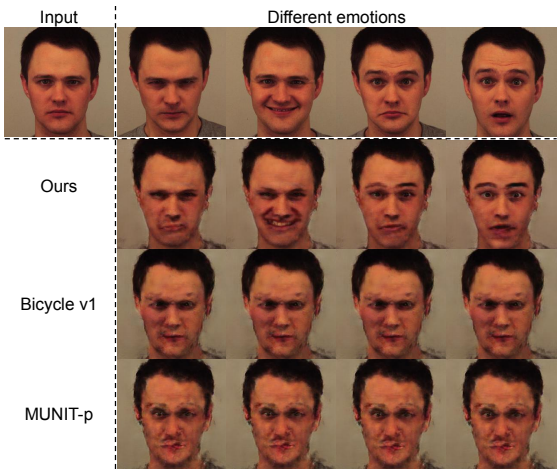


Figure 2.10: Emotion translation results. First row shows the input image, as well as the ground truth images from which we encode the latent emotion vector for reconstruction. Our staged training approach is able to achieve multi-modal synthesis, while the baselines collapse to a single mode.

(2.8c). The embedding learned through pre-training and before fine-tuning shows meaningful clusters. Fine-tuning further brings the embedding closer to that of BicycleGAN.

2.4.12 Encoder pre-training with non-style metrics

Pre-training the encoder using a style-based triplet loss showed to be successful for multi-modal image translation tasks where the variability in the target domain is mainly color-based. This is shown in the results obtained on several benchmarks, even before the fine-tuning stage ("ours - stage 2" in Table 2.1). We note though that the usage of style-loss as a distance metric for triplet sampling is just one choice and can be replaced with other distance metrics depending on the target application. Triplet sampling with style distance results in learning an embedding space where images with similar colors/styles lie closely in that space as shown in § 2.4.11. If, for example, we sample triplets instead based on the distance between VGG-Face [81] embeddings, the encoder will learn a latent space which is clustered by identity. In this section, we aim to validate that the proposed pre-training strategy can be extended to multi-modal image-to-image translation tasks with non-style variability. We inspect the task of manipulating facial expressions, where the input is a neutral face, and the output can have other emotions or facial expressions. For this task, similar emotions should be embedded closely in the latent space. We therefore use an off-the-shelf facial expression recognition system to compute the emotion similarity/distance between any pair of images. Specifically, we compute the emotion distance as the euclidean distance between the 512-dimensional feature map of the last layer of a pre-trained classification network (*e.g.*, [82]). We visualize the learned latent space in Figure 2.9, which shows clusters with similar emotions or facial expressions. We also show example translation re-

sults on a holdout set of the front-view images of the KDEF dataset [83] in Figure 2.10. We note that the generator successfully learns to manipulate facial expressions based solely on the pre-trained embeddings (without the fine-tuning stage). On the other hand, the BicycleGAN-based baselines collapsed to a single mode (over 3 different runs). This shows that our staged-training approach is stable and not sensitive to hyper-parameters, unlike the BicycleGAN baselines which will require careful hyper-parameter tuning to work properly on this task. We also point out that the poor output quality is mainly due to using a pixel-wise reconstruction loss for the generator training, while the input-output pairs in this dataset are not aligned. We did not investigate improving the generator training as this is orthogonal to verifying the generalization of encoder pre-training.

Figure 2.11: Sixteen randomly sampled styles using both the mapper network $\mathcal{M}$ (left), as well as adhoc sampling from the empirically computed $N(\mu, \sigma)$ distribution of a $L2$-regularized latent space (right). Adhoc sampling could sample bad style codes outside the latent distribution (marked in red).

2.4.13 Style sampling comparison

Figure 2.11 compares style sampling using the mapper network $\mathcal{M}$ vs adhoc sampling from the assumed $N(\mu, \sigma)$ of an $L2$-regularized latent space, where μ, σ are empirically computed from the training set. Note that adhoc sampling can sometimes sample bad style codes outside the distribution (*e.g.*, third image in first row, and first image in third row in the right side of Figure 2.11), since the assumption that a $L2$-regularized space would yield normally distributed latents with zero mean and low standard deviation is not explicitly enforced.

2.5 Conclusion

In this chapter, we investigated the effectiveness of embedding pre-training for the task of multi-modal I2I translation. The proposed style-based encoder pre-training (*StEP*) can be done once on auxiliary data, and generalizes well to other domains. This allows for a faster and more stable training of I2I translation networks with fewer losses and achieves more faithful style capture and transfer. Furthermore, we studied the contribution of different loss terms, where we discovered that noise added by a variational auto-encoder can limit the expressiveness of low-dimensional latent spaces. Finally, we achieved state-of-the-art results on several benchmarks.

Chapter 3: Neural Re-rendering in the Wild

In Chapter 2, we explored how to improve multi-modal photo-realistic image syntheis using I2I translation. In the remaining chapters of this dissertation, we extend the application of photo-realistic image synthesis to the task of novel view synthesis, under different challenging settings. We start in this chapter, by exploring total scene capture — recording, modeling, and rerendering a scene under varying appearance such as season and time of day. Starting from internet photos of a tourist landmark, we apply traditional 3D reconstruction to register the photos and approximate the scene as a point cloud. For each photo, we render the scene points into a deep framebuffer, and train a neural network to learn the mapping of these initial renderings to the actual photos. This rerendering network also takes as input a latent appearance vector and a semantic mask indicating the location of transient objects like pedestrians. The model is evaluated on several datasets of publicly available images spanning a broad range of illumination conditions. We create short videos demonstrating realistic manipulation of the image viewpoint, appearance, and semantic labeling. We also compare results with prior work on scene reconstruction from internet photos.

3.1 Introduction

Imagine spending a day sightseeing in Rome in a fully realistic interactive experience without ever stepping on a plane. One could visit the Pantheon in the morning, enjoy the sunset over-

(a) Input deep buffer (b) Output renders

Figure 3.1: Our neural rerendering technique uses a large-scale internet photo collection to reconstruct a proxy 3D model and trains a neural rerendering network that takes as input a deferred-shading deep buffer (consisting of depth, color and semantic labeling) generated from the proxy 3D model (left), and outputs realistic renderings of the scene under multiple appearances (right).

looking the Colosseum, and fight through the crowds to admire the Trevi Fountain at night time. Realizing this goal involves capturing the complete appearance space of a scene, *i.e.* recording a scene under all possible lighting conditions and transient states in which the scene might be observed—be it crowded, rainy, snowy, sunrise, spotlit, etc.—and then being able to summon up any viewpoint of the scene under any such condition. We call this ambitious vision *total scene capture*. It is extremely challenging due to the sheer diversity of appearance—scenes can look dramatically different under night illumination, during special events, or in extreme weather.

In this chapter, we focus on capturing tourist landmarks around the world using publicly available community photos as the sole input, *i.e.* photos *in the wild*. Recent advances in 3D reconstruction can generate impressive 3D models from such photo collections [8, 84, 85], but the renderings produced from the resulting point clouds or meshes lack the realism and diversity of real-world images. Alternatively, one could use webcam footage to record a scene at regular intervals but without viewpoint diversity, or use specialized acquisition (*e.g.*, Google Street View, aerial, or satellite images) to snapshot the environment over a short time window but without ap-

pearance diversity. In contrast, community photos offer an abundant (but challenging) sampling of appearances of a scene over many years.

Our approach to total scene capture has two main components: (1) creating a factored representation of the input images, which separates viewpoint, appearance conditions, and transient objects such as pedestrians, and (2) rendering realistic images from this factored representation. Unlike recent approaches that extract implicit disentangled representations of viewpoint and content [86, 87, 88], we employ state-of-the-art reconstruction methods to create an explicit intermediate 3D representation, in the form of a dense but noisy point cloud, and use this 3D representation as a "scaffolding" to predict images.

An explicit 3D representation lets us cast the rendering problem as a multimodal image translation [2, 42, 43]. The input is a deferred-shading framebuffer [89] in which each rendered pixel stores albedo, depth, and other attributes, and the outputs are realistic views under different appearances. We train the model by generating paired datasets, using the recovered viewpoint parameters of each input image to render a deep buffer of the scene from the same view, *i.e.* with pixelwise alignment. Our model effectively learns to take an approximate initial scene rendering and rerender a realistic image. This is similar to recent neural rerendering frameworks [27, 90] but using uncontrolled internet images rather than carefully captured footage.

We explore a novel strategy to train the multimodal image translation model. Rather than jointly estimating an embedding space for the appearance together with the rendering network [2, 42, 43], our system performs staged training of both. First, an appearance encoding network is pretrained using a proxy style-based loss [75], an efficient way to capture the style of an image. Then, the rerendering network is trained with fixed appearance embeddings from the pretrained encoder. Finally, both the appearance encoding and rerendering networks are jointly finetuned.

This simple yet effective strategy lets us train simpler networks on large datasets. We demonstrate experimentally how a model trained in this fashion better captures scene appearance.

Our system is a first step towards addressing total scene capture and focuses primarily on the static parts of scenes. Transient objects (*e.g.*, pedestrians and cars) are handled by conditioning the rerendering network on the expected semantic labeling of the output image, so that the network can learn to ignore these objects rather than trying to hallucinate their locations. This semantic labeling is also effective at discarding small or thin scene features (*e.g.*, lampposts) whose geometry cannot be robustly reconstructed, yet are easily identified using image segmentation methods. Conditioning our network on a semantic mask also enables the rendering of scenes free of people if desired.

In summary, our contributions include:

- A first step towards total scene capture, *i.e.* recording and rerendering a scene under any appearance from in-the-wild photo collections.
- A factorization of input images into viewpoint, appearance, and semantic labeling, conditioned on an approximate 3D scene proxy, from which we can *rerender* realistic views under varying appearance.
- Compelling results including view and appearance interpolation on five large datasets, and direct comparisons to previous methods [8].

3.2 Related work

Scene reconstruction

Traditional methods for scene reconstruction first generate a sparse reconstruction using large-scale structure-from-motion [84], then perform Multi-View Stereo (MVS) [20, 21] or variational optimization [91] to reconstruct dense scene models. However, most such techniques assume a single appearance, or else simply recover an average appearance of the scene. In this work we build upon these techniques, using dense point clouds recovered from MVS as proxy geometry for neural rerendering.

In image-based rendering (IBR) [92, 93], input images are used to generate new viewpoints by warping input pixels into the outputs using proxy geometry. Recently Hedman *et al.* [94] introduce a neural network to compute blending weights for view-dependent texture mapping that reduces artifacts in poorly reconstructed regions. However, IBR generally assumes static appearance of the captured scene, so it is not well-suited to our problem setup in which the appearance varies across images.

Appearance modeling

A given scene can have dramatically different appearances at different times of day, in different weather conditions, and can also change over the years. Garg *et al.* [95] observe that the dimensionality of scene appearance as captured by internet photos is relatively low given a fixed viewpoint, with the exception of outliers like transient objects. One can recover illumination models for a photo collection by estimating albedo using cloudy images [8], retrieving the

sun's location through timestamps and geolocation [96], or estimating coherent albedos across the collection [97]. However, these methods assume simple lighting models that do not apply to nighttime scene appearance. Radenovic *et al.* [98] recover independent day and night reconstructions, but do not enable smooth appearance interpolations between the two. Sunkavalli *et al.* [99] decompose a time-lapse sequence into sun, sky, shadow, and reflectance components.

Laffont *et al.* [100] assign transient attributes like "fall" or "sunny" to each image, and learn a database of patches that allows for editing such attributes. Other works require direct supervision from lighting models estimated using 360-degree images [101], or ground truth object geometry [102]. In contrast, we use a data-driven implicit representation of appearance that is learned from the input image distribution and does not require direct supervision.

Deep image synthesis

The seminal work of pix2pix [7] trains a deep neural network to translate an image from one domain, such as a semantic labeling, into another domain, such as a realistic image, using paired training data. Image-to-image (I2I) translation has since been utilized for many tasks [22, 25, 30, 31, 32, 73]. Several works proposed improvements to stabilize training and allow for high-quality image synthesis [11, 45, 73]. Others extended the I2I framework to unpaired settings where images from two domains are not in correspondence [66, 67, 70], multimodal outputs where an input image can map to multiple images [2, 45], or unpaired datasets with multimodal outputs where an image in one domain is converted to another domain while preserving the content [42, 43, 74].

Eslami *et al.* [103] introduces a Generative Query Netowrk (GQN) that builds an internal representation of a scene, and uses it to predict the appearance of novel viewpoints. Martin-

(a) initial rendering (b) depth map (c) original image

Figure 3.2: Sample frames of the aligned dataset. Even though interior structures can be seen through the walls in the point cloud rendering (bottom), neural rerendering is able to reason about occlusion among the points and thereby avoid rerendering artifacts. Image credits: James Manners, Patrick Denker (Creative Commons)

Brualla *et al.* [27] propose an image translation framework to fix artifacts in renderings of 3D reconstructions for human performance capture. In this context, a rendered image of a human is *rerendered* using a large convolutional neural network to produce a more realistic output. In this work, we demonstrate an approach for training a neural rerendering framework *in the wild*, *i.e.*, with uncontrolled data instead of captures under constant lighting conditions. We cast this as a multimodal image synthesis problem, where a given viewpoint can be rendered under multiple appearances using a latent appearance vector, and with editable semantics by conditioning the output on the desired semantic labeling of the output.

3.3 Total scene capture

We define the problem of *total scene capture* as creating a generative model for all images of a given scene. We would like such a model to:

- encode the 3D structure of the scene, enabling rendering from an arbitrary viewpoint,
- capture all possible appearances of the scene, *e.g.*, all lighting and weather conditions, and allow rendering the scene under any of them, and
- understand the location and appearance of transient objects in the scene, *e.g.*, pedestrians and cars, and allow for reproducing or omitting them.

Although these goals are ambitious, we show that one can create such a generative model given sufficient images of a scene, such as the ones we can obtain for popular tourist landmarks.

This section presents our approach. We first describe a neural rerendering framework that we adapt from a previous work in controlled capture settings [27] to the more challenging setting of unstructured photo collections (§ 3.3.1). Next, we extend this model to enable appearance capture and multimodal generation of renderings under different appearances (§ 3.3.2). In § 3.3.3 we describe how our model reasons about the location of transient objects in the training data by conditioning its inputs on a semantic labeling of the ground truth images.

3.3.1 Neural rerendering framework

Here we adapt recent neural rerendering frameworks [27, 90] to work with unstructured photo collections. First, we generate a proxy 3D reconstruction of the scene from a large internet photo collection $\{I_i\}$ of the scene, using COLMAP [21, 104, 105]. COLMAP applies Structure-from-Motion (SfM) and Multi-View Stereo (MVS) to create a dense colored point cloud.

An alternative to a point cloud is to generate a textured mesh [106, 107]. Although meshes generate more complete renderings, they tend to also contain pieces of misregistered floating geometry which can occlude large regions of the scene [8]. As we show later, our neural rerendering framework can produce highly realistic images given only point-based renderings as input.

Given the proxy 3D reconstruction, we generate an aligned dataset of rendered images and real images by rendering the 3D point cloud from the viewpoint v_i of each input image I_i, where v_i consists of camera intrinsics and extrinsics recovered via SfM. We generate a deferred-shading deep buffer B_i for each image [89], which may contain per-pixel albedo, normal, depth and any other derivative information. In our case, we only use albedo and depth and render the point cloud by using point splatting with a z-buffer with a radius of 1 pixel, as shown in Figure 3.2.

However, the image-to-image translation paradigm used in [27, 90] is not appropriate for our use case, as it assumes a one-to-one mapping between inputs and outputs. A scene observed from a particular viewpoint can look very different depending on weather, lighting conditions, color balance, post processing filters, etc. In addition, a one-to-one mapping fails to explain transient objects in the scene, such as pedestrians or cars, whose location and individual appearance is impossible to predict from the static scene geometry alone. Interestingly, if one trains a sufficiently large neural network on this simple task, the network learns to a) associate viewpoint with appearance via memorization, and b) hallucinate the location of transient objects, as shown in Figure 3.3.

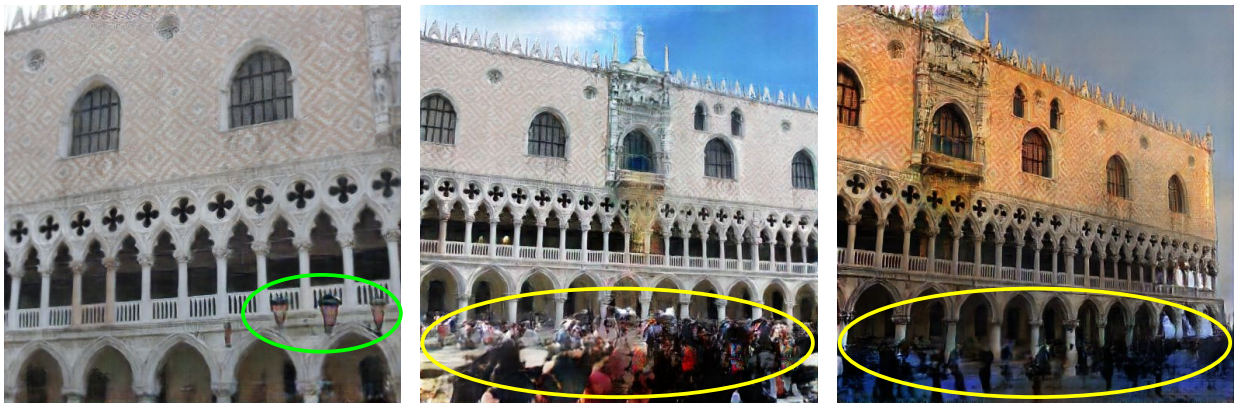


Figure 3.3: Output frames of a standard image translation network [7] trained for neural rerendering using a small dataset of 250 photos of San Marco. The network overfits the dataset and learns to hallucinate lampposts close to their approximate location in the scene (green), and virtual tourists (yellow), as well as memorizing a per-viewpoint appearance matching the specific input photos.

3.3.2 Appearance modeling

To capture the one-to-many relationship between input viewpoints (represented by their deep buffers B_i) and output images I_i under different appearances, we cast the rerendering task as a multimodal image translation [2]. In such a formulation, the goal is to learn a latent appearance vector z_i^a that captures variations in the output domain I_i that cannot be inferred from the input domain B_i. We compute the latent appearance vector as $z_i^a = E^a(I_i, B_i)$ where E^a is an appearance encoder that takes as input both the output image I_i and the deep buffer B_i. We argue that having the appearance encoder E^a observe the input B_i allows it to learn more complex appearance models by correlating the lighting in I_i with scene geometry in B_i. Finally, a rerendering network R generates a scene rendering conditioned on both viewpoint B_i and the latent appearance vector z^a.

To train the appearance encoder E^a and the rendering network R, we first adopt elements from state-of-the-art methods in multimodal synthesis [2, 42, 43] to obtain a combination that

is most effective in our scenario. However, this combination still has shortcomings as it is not able to model less-common appearances well. For instance, it does not reliably capture night appearances for different scenes in our datasets. We hypothesize that the appearance encoder (which is jointly trained with the rendering network) is not expressive enough to capture the large variability in the data.

In order to improve its expressiveness, we consider stabilizing the joint training of R and E^a by pretraining the appearance network E^a independently on a proxy task. We then employ a staged training approach in which the rendering network R is first trained using fixed appearance embeddings, and finally we fine-tune both networks. This staged training regime allows for a simpler model that captures more complex appearances.

Next we present our baseline approach, which adapts state-of-the-art multimodal synthesis techniques, and our staged training strategy which pretrains the appearance encoder on a proxy task.

Baseline. Our baseline is based on BicycleGAN [2] with two main differences. First, our appearance encoder also takes as input the buffer B_i, as described above. Second, we add a cross-cycle consistency loss similar to [42, 43] to encourage appearance transfer across viewpoints. Let $z_1^a = E^a(I_1, B_1)$ be the captured appearance of an input image I_1. We apply a reconstruction loss between image I_1 and a cross-cycle reconstruction $\hat{I}_1 = R(B_1, \hat{z}_1^a)$, where $\hat{z}_1^a$ is computed through a cross-cycle with a second image (I_2, B_2), *i.e.* $\hat{z}_1^a = E^a(R(B_2, z_1^a)), B_2)$. We also apply a GAN loss on the intermediate appearance transfer output $R(B_2, z_1^a)$ as in [42, 43].

Staged appearance training. The key to our staged training approach is the appearance pretrain-

ing stage, where we pretrain the appearance encoder E^a independently on a proxy task. We then train the rendering network R while fixing the weights of E^a, allowing R to find the correlation between output images and the embedding produced by the proxy task. Finally, we fine-tune both E^a and R jointly.

This staged approach simplifies and stabilizes the training of R, enabling training of a simpler network with fewer regularization terms. In particular, we remove the cycle and cross-cycle consistency losses, the latent vector reconstruction loss, and the KL-divergence loss, leaving only a direct reconstruction loss and a GAN loss. We show experimentally in § 3.4 that this approach results in better appearance capture and rerenderings than the baseline model.

Appearance pretraining. To pretrain the appearance encoder E^a, we choose a proxy task that optimizes an embedding of input images into an appearance latent space using a suitable distance metric between input images. This training encourages embeddings such that if two images are close under the distance metric, then their appearance embeddings should also be close in the appearance latent space. Ideally the distance metric we choose should ignore the content or viewpoint of I_i and B_i, as our goal is to encode a latent space that is independent of viewpoint. Experimentally we find that the style loss employed in the neural style-transfer work [75] has such a property; it largely ignores content and focuses on more abstract properties.

To train the embedding, we use a triplet loss, where for each image I_i, we find the set of k closest and furthest neighbor images given by the style loss, from which we can sample a positive sample I_p and negative sample I_n, respectively. The loss is then:

$$\mathcal{L}(I_i, I_p, I_n) = \max\left(\|z_i - z_p\|^2 - \|z_i - z_n\|^2 + \alpha, 0\right)$$

where z_i, z_p, z_n are the latent appearance embedding of I_i, I_p, I_n respectively generated by the appearance encoder E^a, and α is a separation margin.

3.3.3 Semantic conditioning

To account for transient objects in the scene, we condition the rerendering network on a semantic labeling S_i of image I_i that depicts the location of transient objects such as pedestrians. Specifically, we concatenate the semantic labeling S_i to the deep buffer B_i wherever the deep buffer was previously used. This discourages the network from encoding variations caused by the location of transient objects in the appearance vector, or associating such transient objects with specific viewpoints, as shown in Figure 3.3.

A separate benefit of semantic labeling is that it allows the rerendering network to reason about static objects in the scene not captured in the 3D reconstruction, such as lampposts in the San Marco Square. This prevents the network from haphazardly introducing such objects, and instead lets them appear where they are detected in the semantic labeling, which is a significantly simpler task. In addition, by adding the segmentation labeling to the deep buffer, we allow the appearance encoder to reason about semantic categories like sky or ground when computing the appearance latent vector.

We compute "ground truth" semantic segmentations on the input images I_i using DeepLab [108] trained on ADE20K [109]. ADE20K contains 150 classes, which we map to a 3-channel color image. We find that the quality of the semantic labeling is poor on the landmarks themselves, as they contain unique buildings and features, but is reasonable on transient objects.

Using semantic conditioning, the rerendering network takes as input a semantic labeling of

the scene. In order to rerender virtual camera paths, we need to synthesize semantic labelings for each frame in the virtual camera path. To do so, we train a separate semantic labeling network that takes as input the deep buffer B_i, instead of the output image I_i, and estimates a "plausible" semantic labeling $\hat{S}_i$ for that viewpoint given the rendered deep buffer B_i. For simplicity, we train a network with the same architecture as the rendering network (minus the injected appearance vector) on samples (B_i, S_i) from the aligned dataset, and we modify the semantic labelings of the ground truth images S_i to mask out the loss on pixels labeled as transient as defined by a curated list of transient object categories in ADE20K.

3.4 Evaluation

Here we provide an extensive evaluation of our proposed system. Please also refer to the supplementary video[1] to best appreciate the quality of the results from our system.

3.4.1 Implementation details

Our rerendering network is a symmetric encoder-decoder with skip connections, where the generator is adopted from [11] without using progressive growing. We use a multiscale-patchGAN discriminator [45] with 3 scales and employ a LSGAN [57] loss. As a reconstruction loss, we use the perceptual loss [49] evaluated at $conv_{i,2}$ for $i \in [1, 5]$ of VGG [79]. The appearance encoder architecture is adopted from [43], and we use a latent appearance vector $z^a \in \mathbb{R}^8$. We train on 8 GPUs for ~ 40 epochs using 256×256 crops of input images, but we show compelling results on up to 600×900 at test time. The generator runtime for the staged training network is 330 ms for a 512×512 frame on a TitanV without fp16 optimizations. For more

[1] Supplementary video: `y2u.be/E1crWQn_kmY`

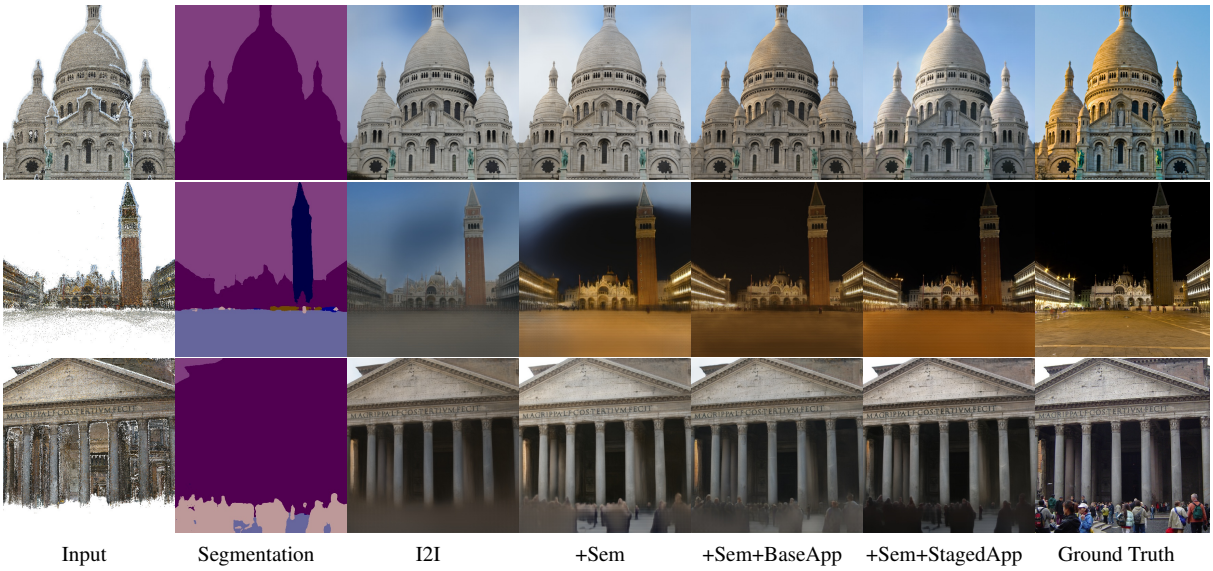


Figure 3.4: Example visual results of our ablative study in Table 3.1. From left to right, input color render, segmentation mask from the corresponding ground truth images, result using an image-to-image baseline (I2I), with semantic conditioning (+Sem), and with semantic conditioning and a baseline appearance modeling based on [2] (+Sem+BaseApp), with semantic conditioning and staged appearance training (+Sem+StagedApp). Photo Credits: Flickr users Gary Campbell-Hall, Steve Collis, and Tahbepet (Creative Commons).

architecture and training details, please refer to § 3.6.

3.4.2 Datasets

We evaluate our method on five datasets reconstructed with COLMAP [104] from public images, summarized in Table 3.1. A separate model is trained for each dataset. We create aligned datasets by rendering the reconstructed point clouds with a minimum dimension of 600 pixels, and throw away sparse renderings (>85% empty pixels), and small images (<450 pixels across). We randomly select a validation set of 100 images per dataset.

Table 3.1: Dataset statistics (number of registered images and size of reconstructed point cloud) and average error on the validation set using VGG/perceptual loss (lower is better), L_1 loss (lower is better), and PSNR (higher is better), for four methods: an image-to-image baseline (I2I), with semantic conditioning (+Sem), with semantic conditioning and a baseline appearance modeling based on [2] (+Sem+BaseApp), and with semantic conditioning and staged appearance training (+Sem+StagedApp).

			I2I			**+Sem**			**+Sem+BaseApp**			**+Sem+StagedApp**		
Dataset	**#Images**	**#Points**	VGG	L_1	PSNR	VGG	L_1	PSNR	VGG	L_1	PSNR	VGG	L_1	PSNR
Sacre Coeur	1165	33M	70.78	39.98	14.36	66.17	34.78	15.62	**60.06**	**21.58**	**18.98**	61.23	25.22	17.81
Trevi	3006	35M	86.52	42.95	14.14	81.82	36.46	15.57	79.10	28.12	17.37	**75.55**	**25.00**	**18.19**
Pantheon	4972	9M	68.28	39.77	14.50	67.47	36.27	15.13	64.06	28.85	16.76	**60.66**	**23.77**	**17.95**
Dubrovnik	5891	33M	78.42	40.60	14.21	78.58	39.88	14.51	76.61	34.57	15.38	**71.65**	**27.48**	**17.01**
San Marco	7711	7M	80.18	44.04	13.97	78.36	39.34	14.58	70.35	26.24	17.87	**68.96**	**23.11**	**18.32**

3.4.3 Ablative study

We perform an ablative study of our system and compare the proposed methods in Figure 3.4. The results of the image-to-image translation baseline method contain additional blurry artifacts near the ground because it hallucinates the locations of pedestrians. Using semantic conditioning, the results improve slightly in those regions. Finally, encoding the appearance of the input photo allows the network to match the appearance. The staged training recovers a closer appearance in San Marco and Pantheon datasets (two bottom rows). However, in Sacre Coeur (top row), the smallest dataset, the baseline appearance model is able to better capture the general appearance of the image, although the staged training model reproduces the directionality of the lighting with more fidelity.

3.4.4 Reconstruction metrics

Table 3.1 reports image reconstruction errors in the validation set using several metrics: perceptual loss [49], L_1 loss, and PSNR. We use the ground truth semantic mask from the source image, and we extract the appearance latent vector using the appearance encoder. Staged training

Figure 3.5: Examples of appearance interpolation for a fixed viewpoint. The left- and rightmost appearances are captured from real images, and the intermediate frames are generated by linearly interpolating the appearances in the latent space. Notice how the baseline method is unable to capture complex scenes, like the sunset and night scene, and its interpolations are rather linear, as can be appreciated in the street lamps (top). The staged training method performs better, but generates twilight artifacts in the sky when interpolating between day and night appearances (bottom).

of the latent appearance space performs better than the baseline for all but the smallest dataset (Sacre Coeur), where the staged training overfits to the training data and is unable to generalize. The baseline method assumes a prior distribution of the latent space and is less prone to overfitting at the cost of poorer modeling of appearance.

3.4.5 Appearance interpolation

The rerendering network allows for interpolating the appearance of two images by interpolating their latent appearance vectors. Figure 3.5 depicts two examples, showing that the staged training approach is able to generate more complex appearance changes, although its generated interpolations lack realism when transitioning between day and night. In the following, we only show results for the staged training model.

Figure 3.6: We capture the appearance of the real images in the left column, and rerender several viewpoints under the captured appearances. The last column is a detail of the previous one. The top row shows the point cloud rendering for the target views, which exhibits artifacts like holes and incomplete features in the statue. Our framework fixes many of the artifacts in the point cloud rendering, adds photo-realistic details, and successfully captures and transfers the source appearnace to the target views. Image credits: Flickr users William Warby, Neil Rickards, Rafael Jimenez, acme401 (Creative Commons).

Figure 3.7: Frames from a synthesized camera path that smoothly transitions from the photo on the left to the photo on the right by smoothly interpolating both viewpoint and the latent appearance vectors. Photo Credits: Allie Caulfield, Tahbepet, Till Westermayer, Elliott Brown.

3.4.6 Appearance transfer

Figure 3.6 demonstrates the ability of our full model to transfer the appearance of a given photo to others. It shows realistic renderings of the Trevi fountain from five different viewpoints under four different appearances obtained from other photos. Note the sunny highlights and the spotlit night illumination appearance of the statues. However, these details can flicker when synthesizing a smooth camera path or smoothly interpolating the appearance in the latent space, as seen in the supplementary video, since our training does not enforce temporal consistency.

3.4.7 Image interpolation

Figure 3.7 shows sets of two images and frames of smooth image interpolations between them, where both the viewpoint and appearance transition smoothly between them. Note how the illumination of the scene can transition smoothly from night to day.

(a) w/ GT segmentation (b) w/ predicted segmentations

Figure 3.8: Example semantic labelings and output renders when using the "ground truth" segmentation mask computed from the corresponding real image (from the validation set) and the predicted one from the associated deep buffer. Note the artifacts on the bottom right where the ground is misclassified as building.

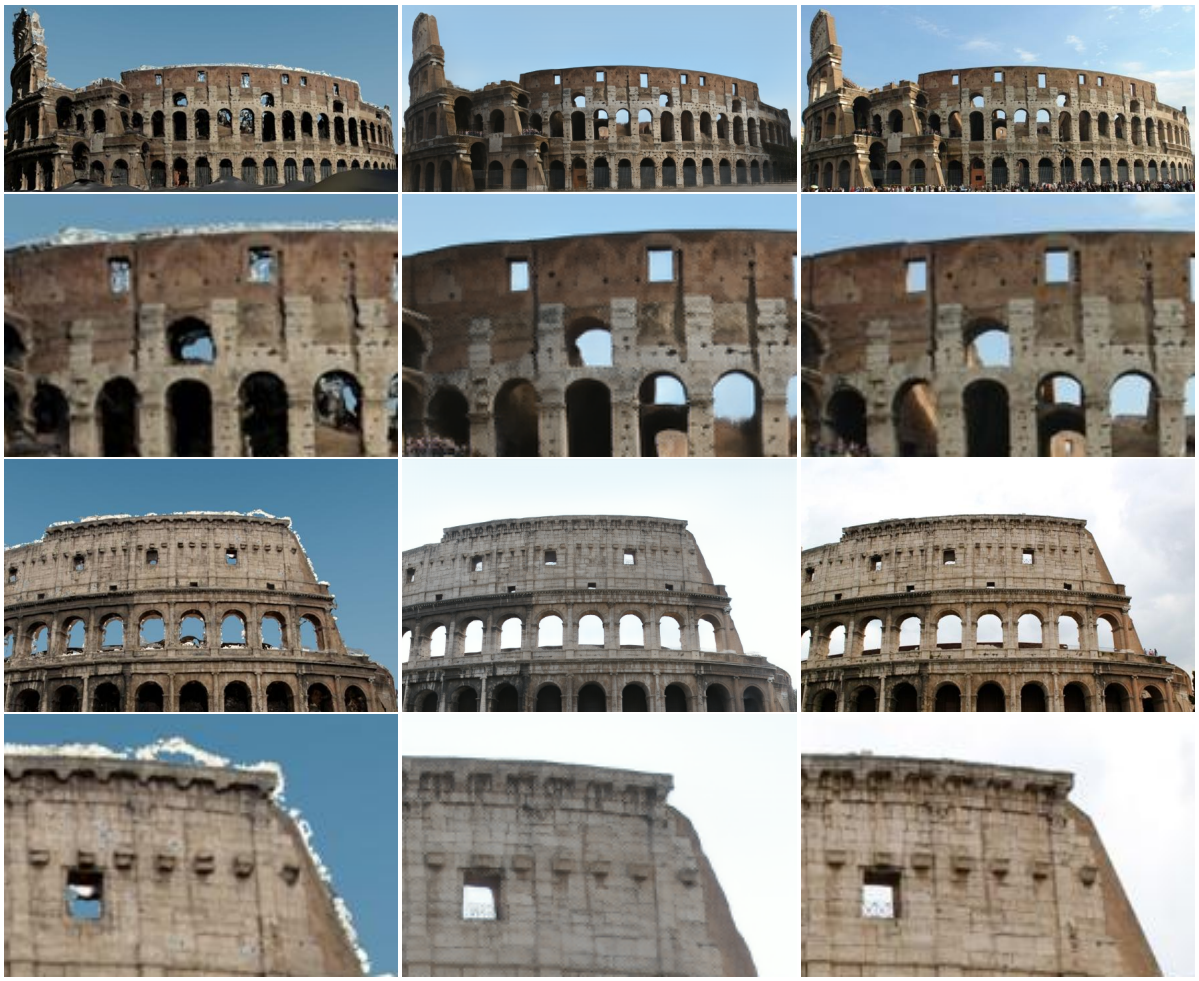
3.4.8 Semantic consistency

Figure 3.8 shows the output of the staged training model with ground truth and predicted segmentation masks. Using predicted segemntation masks, the network produces similar results on the building, but is able to render a scene free of people. Note however how the network depicts pedestrians as black, ghostly figures when they appear in the segmentation mask.

3.4.9 Comparison to 3D reconstruction methods

We evaluate our technique against the one of Shan *et al.* [8] on the Colosseum, which contains 3K images, 10M color vertices and 48M triangles and was generated from Flickr, Google Street View, and aerial images. Their 3D representation is a dense vertex-colored mesh, where the albedo and vertex normals are jointly recovered together with a simple 8-dimensional lighting model (diffuse, plus directional lighting) for each image in the photo collection.

Figure 3.9 compares both methods and the original ground truth image. Their method suffers from floating white geometry on the top edge of the Colosseum, and has less detail, although

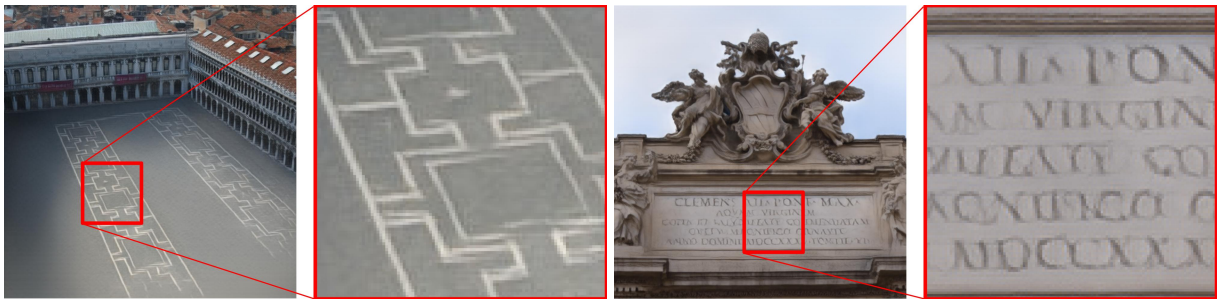


(a) [8] (b) ours (c) original images

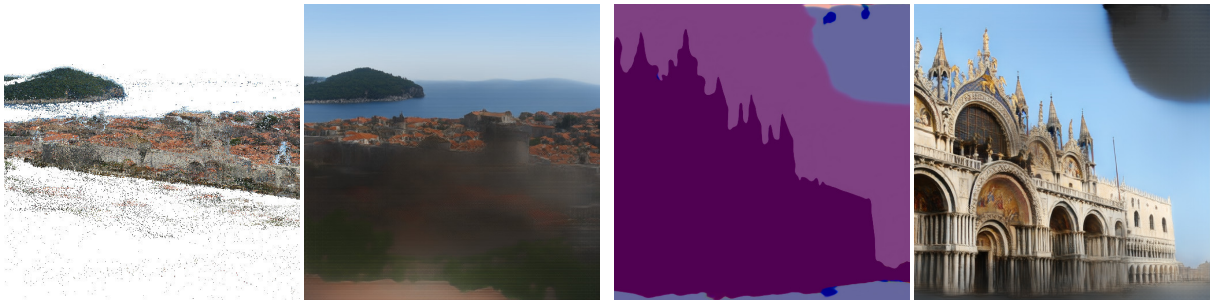
Figure 3.9: Comparison of [8] and our approach. Rows 1 & 3: original photos. Rows 2 & 4: detailed crops. Image credits: Graeme Churchard, Sarah-Rose (Creative Commons).

it recovers the lighting better than our method, thanks to its explicit lighting reasoning. Note that both models are accessing the test image to compute lighting coefficients and appearance latent vectors, with dimension 8 in both cases, and that we use the predicted segmentation labelings from B_i.

We ran a randomized user study on 20 random sets of output images that do not contain close-ups of people or cars, and were not in our training set. For each viewpoint, 200 participants chose "which image looks most real?" between an output of their system and ours (without seeing



(a) Neural artifacts



(b) Sparse reconstructions

(c) Segmentations artifacts

Figure 3.10: Limitations of the current system, described in Section 3.4.10.

the original). Respondents preferred images generated by our system a 69.9% of the time, with our technique being preferred on all but one of the images. We show many more comparisons in § 3.5.2.

3.4.10 Limitations

Our system has several limitations, that are significantly different from those of traditional 3D reconstruction pipelines:

- **Segmentation:** Our model relies heavily on the segmentation mask to synthesize parts of the image not modeled in the proxy geometry, like the ground or sky regions. Thus our results are very sensitive to errors in the segmentation network, like in the sky region in Figure 3.10c or an appearing "ghost pole" artifact in San Marco (frame 40 of bottom row in Figure 3.7, best seen in video). Jointly training the neural rerenderer together with the

segmentation network could reduce such artifacts.

- **Neural artifacts:** Neural networks are known to produce screen door patterns [110] and other intriguing artifacts [111]. We observe evidence of such artifacts in repeated structures, like the patterns on the floor of San Marco, that in our renderings don't line up, as if they had been hand painted. Similarly, the inscription on top of the Trevi fountain is reproduced with a distorted font (see Figure 3.10a).

- **Incomplete reconstructions:** Sometimes an image contains partially reconstructed parts of the 3D model, creating large holes in the rendered B_i. This forces the network to hallucinate the incomplete regions, generally leading to blurry outputs, as seen in Figure 3.10b.

- **Temporal artifacts:** When smoothly varying the viewpoint, sometimes the appearance of the scene can flicker considerably, especially under complex appearance, such as when the sun hits the Trevi Fountain, creating complex highlights and cast shadows. Please see the supplementary video for an example.

3.5 Additional results

3.5.1 Appearance variation

In this section, we extend Figure 3.6 and include additional results for diverse appearances modeled by our proposed staged training method on the San Marco dataset. Figure 3.11 shows realistic renderings of five different scenes/viewpoints of San Marco under four different appearances obtained from other photos.

3.5.2 Qualitative comparison

We evaluated our technique against Shan *et al.* [8] on the Colosseum. In § 3.4.9, we reported the result of a user study run on 20 randomly selected sets of output images that do not contain close-ups of people or cars, and were not in our training set. Figures 3.12, 3.13 show a side-by-side comparison of all 20 images used in the user study.

Table 3.2: Re-evaluating our proposed method with using estimated segmentation masks as opposed to Table 3.1, which uses segmentation masks computed from the ground truth images.

	+Sem+StagedApp		
Dataset	VGG	L_1	PSNR
Sacre Coeur	67.74	28.66	16.45
Trevi	77.35	26.03	17.90
Pantheon	62.54	25.40	17.29
Dubrovnik	74.44	30.39	16.18
San Marco	75.58	26.69	17.34

3.5.3 Quantitative evaluation with learned segmentations

To quantitatively evaluate rerendering using estimated segmentation masks, we generate plausible semantic masks for the validation set, as described in § 3.3.3, and recompute the quantitative metrics, as in Table 3.1, for our proposed method. Note that estimated semantic maps will not perfectly match those of the ground truth validation images. For example, ground truth semantic maps could contain the segmentation of transient objects, like people or trees. So, it is not fair to compare reconstructions based on estimated segmentation maps to the ground truth validation images. While results in Table 3.2 show some performance drop as expected, we still get a reasonable performance compared to that in Table 3.1. In fact, we still perform better than the BicycleGAN baseline on the Trevi, Pantheon and Dubrobnik datasets, even though the BicycleGAN baseline uses ground truth segmentation masks.

Figure 3.11: We capture the appearance of the original images in the first row, and rerender several viewpoints under them. The first column shows the rendered point cloud images used as input to the rerenderer.

Figure 3.12: Comparison with Shan *et al.* [8] – set 1 of 2. First and third columns show the result of Shan *et al.* [8]. Second and fourth columns show our result.

Figure 3.13: Comparison with Shan *et al.* [8] – set 2 of 2. First and third columns show the result of Shan *et al.* [8]. Second and fourth columns show our result.

3.6 Additional implementation details

We used different networks for the staged training and the baseline models. We obtained best results for each model with different networks. Below, we provide an overview of the different architectures used in the staged training and the baseline models. For more details, please refer to our publicly released code.

3.6.1 Neural rerender network architecture

Our rerendering network is a symmetric encoder-decoder with skip connections. The generator is adopted from [11] without using progressive growing. Specifically, we extend the GAN architecture in [11] to a conditional GAN setting. The encoder/decoder operates at a 256×256 resolution, with 6 downsampling/upsampling blocks. Each block has a downsampling/upsampling layer followed by two single-strided 3×3 *conv* layers with a *leaky ReLu* ($\alpha = 0.2$) and *pixel-norm* [11] layers. We add skip connections between the encoder and decoder by concatenating feature maps at the beginning of each decoder block. We use 64 feature maps at the first encoder block and double the size of feature maps after each downsampling layer until it reaches size 512.

3.6.2 Appearance encoder architecture

We implemented the appearance encoder architecture used in [43] except that we added *pixel-norm* [11] layers after each downsampling block. We observed that adding a pixel-wise normalization layer stabilizes the training while at the same time avoids mixing information between different pixels as in *instance norm* or *batch norm*. We use a latent appearance vector

$z^a \in \mathbb{R}^8$. The latent vector is injected at the bottleneck between the encoder and decoder in the rendering network. We tile z^a to match the dimension of feature maps at the bottleneck and concatenate it to the feature maps channel-wise.

3.6.3 Baseline architecture

We used a faithful Tensorflow implementation of the encoder-decoder network and appearance encoder in [43] using their PyTorch released code as a guideline. We adapted their training pipeline to the single-domain supervised setup as described in § 3.3.2.

3.7 Conclusion

In this chapter, we presented a first attempt at solving the total scene capture problem. Using unstructured internet photos, we proposed a neural rerendering framework that is able to produce highly realistic scenes under different illumination conditions. The proposed framework scales up novel view synthesis to data in the wild by constructing a point cloud as a proxy geometry for the scene, and augmenting it with an I2I translation network to map rough point cloud renders to photo-realistic images. We leveraged the style pre-training strategy we proposed in Chapter 2 to better capture the appearance of the scene as seen in internet photos. Finally, we evaluated our system on five challenging datasets and against state-of-the-art 3D reconstruction methods.

Chapter 4: LSR: Learned Spatial Representations for Few-shot Talking-Head Synthesis

In Chapter 3, we tackled novel view synthesis from unstructured internet photos, where we have thousands of images of some static structure (*e.g.*, a tourist landmark), but images contain transient occluders, and they exhibit extreme variation in appearance. In this chapter, we move to an even more challenging setting of novel view synthesis, where the target scene is a dynamic (moving) human subject, and we only have a handful or even a single image for that subject. To this end, we propose *LSR*, a framework for few-shot talking-head synthesis. While recent works on neural talking heads have produced promising results, they can still produce images that do not preserve the identity of the subject in source images. We posit this is a result of the entangled representation of each subject in a single latent code that models 3D shape information, identity cues, colors, lighting and even background details. In contrast, we propose to factorize the representation of a subject into its spatial and style components. Our method generates a target frame in two steps. First, it predicts a dense spatial layout for the target image. Second, an image generator utilizes the predicted layout for spatial denormalization and synthesizes the target frame. We experimentally show that this disentangled representation leads to a significant improvement over previous methods, both quantitatively and qualitatively.

Figure 4.1: Our framework disentangles spatial and style information for image synthesis. It predicts a latent spatial layout for the target image, which is used to produce per-pixel style modulation parameters for the final synthesis.

4.1 Introduction

We study the task of learning personalized head avatars in a low-shot setting, also known as *"neural talking heads"*. Given a single-shot or few-shot images of a source subject, and a driving sequence of facial landmarks, possibly derived from a different subject, the goal is to synthesize a photo-realistic video of the source subject, under the poses and expressions of the driving sequence. This task has a wide range of applications, including those in AR/VR, video conferencing, gaming, animated movie production and video compression in tele-communication.

Traditional graphics-based approaches to this task rely on a 3D face geometry and produce very high quality synthesis. However, they tend to focus on modeling the face area without the hair, and they learn a subject-specific model and cannot generalize to new subjects. In contrast, recent 2D-based approaches [3, 112, 113, 114] learn a subject-agnostic model that can animate unseen subjects given as few as a single image. Furthermore, since these works learn an implicit model and do not require an explicit geometric representation, they can synthesize the full head, including the hair, mouth interior, and even wearable accessories like glasses and earrings. This remarkable generalization ability however comes at the cost of low quality and poor identity preservation when compared to their 3D-based subject-specific counterparts. Bridging the quality

gap between 2D-based subject-agnostic and 3D-based subject-specific approaches remains an open problem.

Recent efforts in 2D-based approaches can be divided into two classes; *warping-based* and *direct synthesis*. As the name suggests, warping-based methods (*e.g.*, [3]) learn to warp the input image or a recovered canonical pose based on the motion of the driving sequence. While these methods achieve high realism, especially for static and rigid parts of the image, they tend to work well only for a limited range of motion, head rotation and dis-occlusion. On the other hand, direct synthesis approaches (*e.g.*, [112, 113, 114]) encode the source subject into a compressed latent code, and a generator decodes the latent code to synthesize the target pose. These approaches learn a prior over the compressed latent space, and can generate realistic results for a wider range of poses and head motion. However, they exhibit a noticeable identity gap between their output and the source subject.

We posit that the identity gap is caused by the entangled representation of the source subject in a single latent code. This compressed 1D latent encodes multi-view shape information, identity cues, as well as color information, lighting and background details. In order to synthesize a target view from a latent code, the generator needs to devise a complex function to decode the uni-dimensional latent into its corresponding 2D spatial information. We argue this not only consumes a large portion of the network capacity, but also limits the amount of information that can be encoded in the latent code.

To address this problem, we propose a two-step framework that decomposes the synthesis of a talking head into its spatial and style components. Our framework animates a source subject in two steps. First, it predicts a novel spatial layout of the subject under the target pose and expression. Then, it synthesizes the target frame conditioned on the predicted layout. This

factorized representation yields the following key performance advantages.

- **Better subject-agnostic model performance.** The performance of our subject-agnostic (also called *meta-learned*) model not only performs better than previous subject-agnostic state-of-the-art, but is also on-par with the subject-finetuned performance of previous works when there are only few source images available (*e.g.*, less than 10 images).

- **Better fine-tuned performance with less data.** Fine-tuning our model for a specific subject requires significantly less data and fewer iterations than previous works, and yet achieves better performance. For example, we show that fine-tuning our model using 4-shot inputs outperforms previous state-of-the-art models fine-tuned using 32-shot inputs.

- **Robustness to pose variations.** We show that our model is more robust against a wider range of poses and facial expressions, while still producing both realistic and identity-preserving results.

- **Improved identity preservation.** Shape difference between the source and driving identities poses a challenge for identity preservation in reenacted results. The intermediate novel spatial representation learned by our model reduces the sensitivity towards such differences and better preserves the identity.

In summary, we make the following contributions:

- A novel approach that disentangles the spatial and style components for *talking-head* synthesis.

- A novel latent spatial representation that proves effective for few-shot novel view synthesis.

- We achieve state-of-the-art performance in both the single-shot and multi-shot settings, as well as in the meta-learned and subject-finetuned modes.

4.2 Related work

Existing approaches for realistic talking-head synthesis can be categorized into 3D-based and 2D-based.

3D-based methods. Such methods [115, 116, 117] utilize 3D geometric representations as a proxy to animate a target subject. Common geometric representations, such as 3D morphable models (3DMM) [118], only model the face area, and do not include challenging regions like the hair, eyes and mouth interior. Obtaining a detailed geometry of these regions is an expensive and challenging task. Therefore, such methods either cannot synthesize or perform poorly on those regions. Recent works [29, 90, 119] combine the traditional graphics pipeline with machine learning to better model the eye movement, mouth interior, or learn a better appearance model. However, they learn subject-specific models that do not generalize to new subjects. Other works [120, 121] take first steps to generalize to multiple subjects but they do not perform well on hair and other regions outside the face.

2D-based methods. These methods [3, 112, 113, 114, 122, 123, 124, 125, 126, 127] learn an implicit model of the head and do not require a proxy geometry. Therefore they can synthesize the full head including dynamic regions like the hair, eyes, and mouth interior. They can also model different wearable accessories such as hats, glasses, and earrings. Early works

build on top of CycleGAN [70] and learn subject-specific models [128, 129]. More recent works [3, 112, 113, 114, 126, 127] learn subject agnostic models that can animate unseen subjects given only a single or few-shot images. However, these methods lack in quality and identity preservation compared to the 3D-based subject-specific models. To bridge this performance gap, hybrid models [112, 113, 114] utilize a meta-learning phase that trains a subject-agnostic model on a large corpus of data, then an optional subject-specific fine-tuning phase is performed to improve the realism and restore the source identity. In this work, we improve the meta-learned performance to achieve state-of-the-art results without any subject-specific fine-tuning. While our model could still benefit from the optional fine-tuning phase to further refine the results, it requires significantly less data samples compared to previous works.

On another axis, 2D-based approaches can be categorized based on the synthesis technique into warping-based (*e.g.*, [3, 126, 130, 131]) and direct synthesis (*e.g.*, [112, 113, 114]). Warping-based approaches warp an input image [3, 126] or a recovered canonical pose [130] to synthesize novel poses. Warping results however tend to break when the target pose is far from that of the source image. Direct synthesis approaches utilize advances in Generative Adversarial Networks (GANs) [5] and Image-to-Image (I2I) translation [7] to generate novel poses. Compared to warping-based approaches, direct synthesis methods can realistically handle a wider range of poses and expressions.

Multi-modal Image-to-Image (I2I) translation. Several multi-modal I2I translation works feed a style latent code, either directly to the generator [2] or through adaptive instance normalization (AdaIN) [42, 132]. Recent state-of-the-art architectures [133, 134, 135] showed a significant improvement over traditional UNet [136] and encoder-decoder architectures, by generating per-

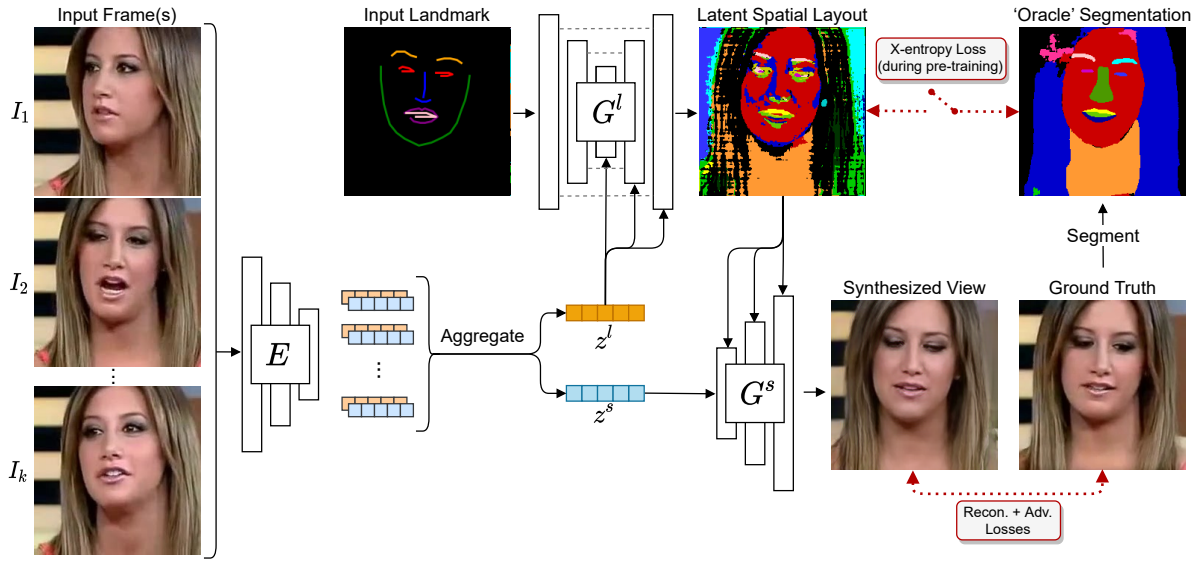


Figure 4.2: Overview of our training pipeline. The cross-entropy loss with the oracle segmentation is only used during pre-training the layout predictor G^l, and then turned off during the full pipeline training.

pixel spatial denormalization (SPADE) parameters [133]. However, such architectures depend on the existence of accurate semantic segmentations or other dense spatial representations of the target image, hence limiting their usage in tasks where such dense representations do not exist. In this work, we learn to predict a latent dense layout to provide the spatial input to SPADE.

4.3 Method

Our approach factorizes the representation of a head avatar into spatial and style components. It breaks down the novel view head synthesis of a subject into two steps. First, a layout prediction network G^l translates facial landmarks for a target view into a dense spatial layout of the subject. Then, an image generator G^s synthesizes the final image conditioned on the predicted layout. We first give an overview of our pipeline in § 4.3.1. Then, we explain how to pre-train the layout prediction network G^l to predict semantic segmentations of novel views in § 4.3.2, followed by the full pipeline training in § 4.3.3. § 4.3.4 explains how the layout pre-

diction network G^l transitions from predicting semantic maps to learning a more powerful latent spatial representation. And finally, we discuss how to learn a personalized head avatar through an optional subject-specific fine-tuning stage in § 4.3.5.

4.3.1 Overview

Given K-shot inputs $\{I_1 \dots I_K\}$ of a source subject, a two-headed encoder $E = \{E^l, E^s\}$ processes the inputs and generates K layout latents $\{z_i^l\}$ and K style latents $\{z_i^s\}$ for $i \in \{1 \dots K\}$. The K latents are then averaged to get an aggregated layout latent $z^l = \frac{1}{K}\sum_{i=1}^{K} z_i^l$ and style latent $z^s = \frac{1}{K}\sum_{i=1}^{K} z_i^s$. Averaging the K latents cancels out view-specific information and transient occluders, and maintains implicit 3D information like the head and hair shape for the layout latent, and color and lighting information for the style latent. We have two generators: a layout predictor network G^l and an image generator G^s. The layout predictor takes as input the facial landmarks for a target view x_t and the layout latent z^l and generates a spatial layout $y_t^l = G^l(x_t, z^l)$, such as a semantic map, for the target view. The image generator G^s processes the style latent z^s and utilizes spatial denormalization layers (SPADE [133]), conditioned on the predicted layout y^l, to synthesize the final image $\hat{I}_t = G^s(y_t^l, z^s)$. An overview of our framework is shown in Figure 4.2.

4.3.2 Layout prediction pre-training

Training the above pipeline end-to-end without any supervision or constraints on the predicted layouts results in a degenerate solution, where the spatial layouts and their corresponding spatial denormalization are completely ignored. All spatial and style information are thus en-

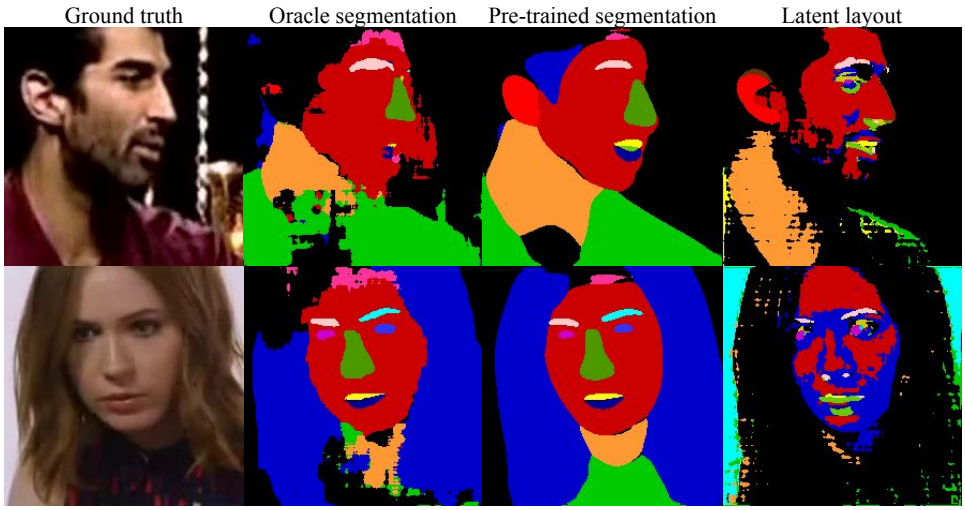


Figure 4.3: Layout pre-training predicts meaningful segmentation maps despite the noisy oracle segmentations. Our latent spatial representation encodes more information than traditional segmentations.

coded into and decoded from the style latent z^s, which results in a poor performance. Therefore, we opted to pre-train the layout prediction network to predict a plausible semantic segmentation of a target view, given the input facial landmarks x_t and the layout latent z^l. To supervise this training, we use an off-the-shelf face segmentation network [137] as an oracle to segment the target image I_t into a semantic map S_t, and we apply a cross-entropy loss (X-ent) between the oracle segmentation S_t and our predicted segmentation $y_t^l = G^l(x_t, z^l)$. We observe that the obtained oracle segmentations are very noisy and have poor quality (*e.g.*, Figure 4.3). This is caused by the domain gap, in terms of image resolution and the distribution of head poses, between the datasets used to train the oracle segmentation network [137], and in-the-wild videos of talking heads. Thus, to regularize the segmentation prediction training, we use a mutli-task pre-training strategy where the layout prediction network predicts an extra RGB reconstruction R_t of the target image I_t, which is used as a secondary supervisory signal. Specifically, we have

$$y_t^l, R_t = G^l(x_t, z^l), \qquad z^l = \frac{1}{K}\sum_{i=1}^{K} E^l(I_i) \tag{4.1}$$

And the objective for the pre-training is

$$\mathcal{L}_{\text{seg}} = \text{X-ent}(y_t^l, S_t) + \lambda_R \mathcal{L}_R(R_t, I_t) \tag{4.2}$$

where $\mathcal{L}_R$ is a perceptual reconstruction loss, and λ_R is a relative weighting term which is set to a low value.

4.3.3 Full pipeline training

Once the layout predictor network has been pre-trained to predict semantic segmentations, we plug it into our full pipeline. The predicted segmentation is fed as the spatial input to a SPADE image generator G^s that synthesizes the final image as

$$\hat{I}_t = G^s(G^l(x_t, z^l), z^s), \qquad z^s = \frac{1}{K}\sum_{i=1}^{K} E^s(I_i) \tag{4.3}$$

We observe that the SPADE generator quickly utilizes the input spatial segmentations to resolve spatial ambiguities, and we no longer fall into a degenerate solution where the spatial input is ignored.

Our full pipeline, comprising the layout and style encoders $\{E^l, E^s\}$, the layout predictor G^l and the image generator G^s, is optimized to minimize three losses; a reconstruction loss $\mathcal{L}_{\text{rec}}$, an adversarial loss $\mathcal{L}_{\text{adv}}$, and a latent regularization loss $\mathcal{L}_{L2}$.

For the reconstruction loss $\mathcal{L}_{\text{rec}}$, we employ a perceptual loss [49] based on both the VGG19 [79] and VGGFace [81] networks, as well as an $L1$ loss. While the VGG19-based perceptual loss is a standard reconstruction loss, we follow Zakharov *et al.* [112] and utilize a VGGFace-based perceptual loss to promote identity preservation. We also use an $L1$ loss to better preserve color transfer between the synthesized and ground truth images.

The adversarial loss, $\mathcal{L}_{\text{adv}}$, encourages the output to be photo-realistic. To achieve that, a

discriminator network D is trained to discriminate between real and fake images, while the generator network, G^s aims to fool the discriminator by bringing the output closer to the manifold of real images. We borrow the architecture of the discriminator network D from [14] and use a non-saturating logistic loss with gradient penalty [138]. Finally, we impose an $L2$ regularization on the learned latent codes to encourage compactness of the latent space. The full training objective is given by

$$\begin{aligned} \min \mathcal{L}(\hat{I}_t, I_t, z^l, z^s | E^l, E^s, G^l, G^s, D) = \mathcal{L}_{\text{rec}}(\hat{I}_t, I_t) + \\ \lambda_{\text{adv}} \mathcal{L}_{\text{adv}}(\hat{I}_t, I_t) + \lambda_{L2} \big(\|z^l\|^2 + \|z^s\|^2 \big) \end{aligned} \tag{4.4}$$

where $\lambda_{\text{rec}}, \lambda_{L2}$ determine the relative weights between the loss terms.

4.3.4 Learning a latent spatial representation

Spatial denormalization (SPADE) generates per-pixel denormalization parameters by feeding a dense spatial input through a small convolutional subnetwork. While SPADE [133] originally uses semantic maps as input, we explore learning a latent spatial representation that better suits the image synthesis task at hand. To do this, we turn off the cross-entropy loss so as to give the layout predictor G^l the freedom to diverge from predicting traditional semantic segmentations and learn other latent representations that better optimize the few-shot novel view synthesis objective. The layout predictor is thus supervised only by the training objective of Eqn. 4.4. Figure 4.3 shows examples of the learned latent layouts. Although they might look less interpretable than traditional semantic maps, they seem to encode more information and capture accurate details.

4.3.5 Subject fine-tuning

Training our full pipeline learns a powerful subject-agnostic model that produces high quality and identity-preserving synthesis. Optionally, we can learn a personalized head avatar to further refine the results for a given subject. To do this, we follow [112, 113, 114] and fine-tune the subject-agnostic model (also called *meta-learned* model) using the few-shot inputs of the source identity. Specifically, we compute the layout and style embeddings $\{z^l, z^s\}$ and fine-tune the weights of the layout and image generators $\{G^l, G^s\}$, as well as the discriminator, D, by reconstructing the set of few-shot inputs, and optimizing the same training objective of Eqn. 4.4. We observe that subject fine-tuning restores high-frequency components and improves background reconstruction when compared to the meta-learned outputs.

4.4 Experimental evaluation

Implementation details. Please, refer to § 4.6 for networks architecture, hyper-parameters and training details.

Dataset. We perform our evaluation using the VoxCeleb [139] dataset, which is a large-scale in-the-wild video dataset. The train set contains over a million clips from 145,569 videos of 5,994 different identities. The test set contains new identities that are not part of the training. We use the test subset released by Zakharov *et al.* [112], which contains a total of 1,600 frames from videos of 50 subjects. For self-reenactment scenarios, the input few-shots and the driving sequence do not overlap. We obtain the facial landmarks for sampled frames using an off-the-shelf facial landmarks detector [140].

Baselines. We compare our method to the following baselines: X2Face [130], FSTH [112], FOMM [3], Latent Pose Descriptor (LPD) [113], and Bi-layer [114]. We use the released pre-trained models provided by the authors for all baselines, except for FSTH [112] where we use the authors' provided outputs, as their code and models were not released. Since some baselines only accept single-shot inputs (*e.g.*, FOMM and Bi-layer), we divide our evaluation into a single-shot setting, where we compare to all the baselines, and a multi-shot setting, where we only compare against the few-shot baselines. Since the LPD [113] and Bi-layer [114] baselines do not predict the background and re-crop the input/output frames, we subtract the background and compare with their corresponding cropped ground truths for quantitative analysis. We also exclude those two baselines from frame reconstruction evaluation since their output does not align with the rest of the methods.

Metrics. We evaluate all models along five axes.

- Reconstruction fidelity using the peak signal-to-noise ratio (*PSNR*) and structural similarity (*SSIM*) [141] metrics.
- Perceptual similarity between the output and the ground truth using the *AlexNet*-based *LPIPS* metric [1].
- Identity preservation (*ID-SIM*) using the cosine similarity between face embeddings from a face recognition network [81].
- Normalized Mean Keypoint Error (*NMKE*), which measures the pose error between the synthesized and ground truth images as computed in [113, 114].

Table 4.1: Quantitative comparison in the single-shot setting.

Method	PSNR↑	SSIM↑	LPIPS↓	ID-SIM↑	NMKE↓	FID↓
X2Face [130]	15.50	0.466	0.346	0.691	0.333	98.58
Bi-layer [114]	–	–	–	0.721	0.236	130.58
FSTH [112]	16.92	0.597	0.263	0.836	0.049	53.07
LPD [113]	–	–	–	0.837	0.070	48.48
FOMM [3]	**18.20**	**0.635**	0.236	0.869	0.061	56.10
Ours	17.37	0.605	**0.232**	**0.886**	**0.041**	**45.69**

- Perceptual quality of the output using the Frechet-Inception Distance (*FID*) metric [142].

4.4.1 Single-shot comparative evaluation

Table 4.1 shows a quantitative comparison with the baselines in the single-shot setting. Our method outperforms all baselines in perceptual reconstruction (LPIPS), identity preservation (ID-SIM), pose matching (NMKE) and visual quality (FID). However, FOMM scores better in the standard reconstruction metrics (PSNR and SSIM). We argue this is intrinsic to their method due to its warping-based nature, which accurately captures the background and other static regions, and thus gives low reconstruction error even in the presence of clear artifacts. Furthermore, while FOMM cannot utilize more input frames to its advantage, our method's performance improves with multi-shot inputs to significantly surpass FOMM in all metrics (refer to § 4.5 for the quantitative numbers).

Figure 4.4 shows qualitative results from three groups representing low, medium and high variance between the input and target poses. We observe that all methods perform well when the target pose is similar to that of the input shot. LPD produces sharp results within the low-medium pose variation, but shows blurry artifacts within the face and eyes in the case of high pose variance. FSTH shows a clear identity gap. FOMM accurately matches the background and

Figure 4.4: Qualitative comparison in the single-shot setting. We show three sets of examples representing low, medium and high variance between the source and target poses. Our method is more robust to pose variations than the baselines.

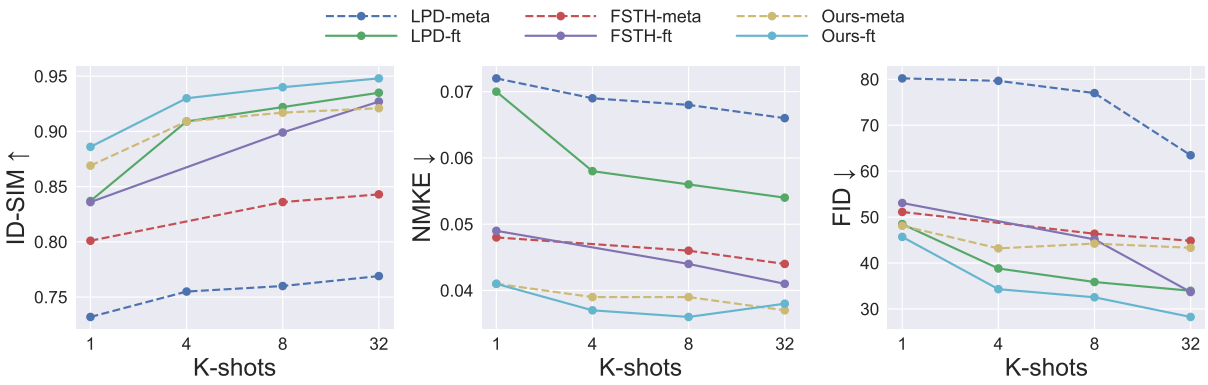


Figure 4.5: Quantitative comparison with the few-shot baselines, showing the effect of both increasing the K-shot inputs and subject-specific fine-tuning. Dotted and solid lines represent the meta-learned and fine-tuned models respectively.

shows highly realistic results when the pose variance is low, but shows a clear identity gap and visible artifacts when the target pose is far from the source image. Our method is more robust against pose variation, yielding realistic results while preserving the source identity.

4.4.2 Multi-shot comparative evaluation

Here, we focus on the effect of increasing the number of K-shot inputs, and the effect of subject-specific fine-tuning using the K-shot inputs. Figure 4.5 plots the ID-SIM, NMKE and FID performance metrics as we increase the number of K-shots. We observe that the pose reconstruction performance (NMKE) is mainly dictated by the approach itself, rather than the number of K-shots or whether the models are fine-tuned or not. For example, the *meta-learning* performance of FSTH with $K = 1$ is better than the *fine-tuned* LPD model with $K = 32$. Similarly, the *single-shot meta-learning* performance of our method is better than the *fine-tuned* baselines at $K = 32$.

For the ID-SIM and FID metrics, the *meta-learning* performance of our model is not only superior to that of the baselines, but it is also on-par with the *fine-tuned* baselines for $K \leq 8$.

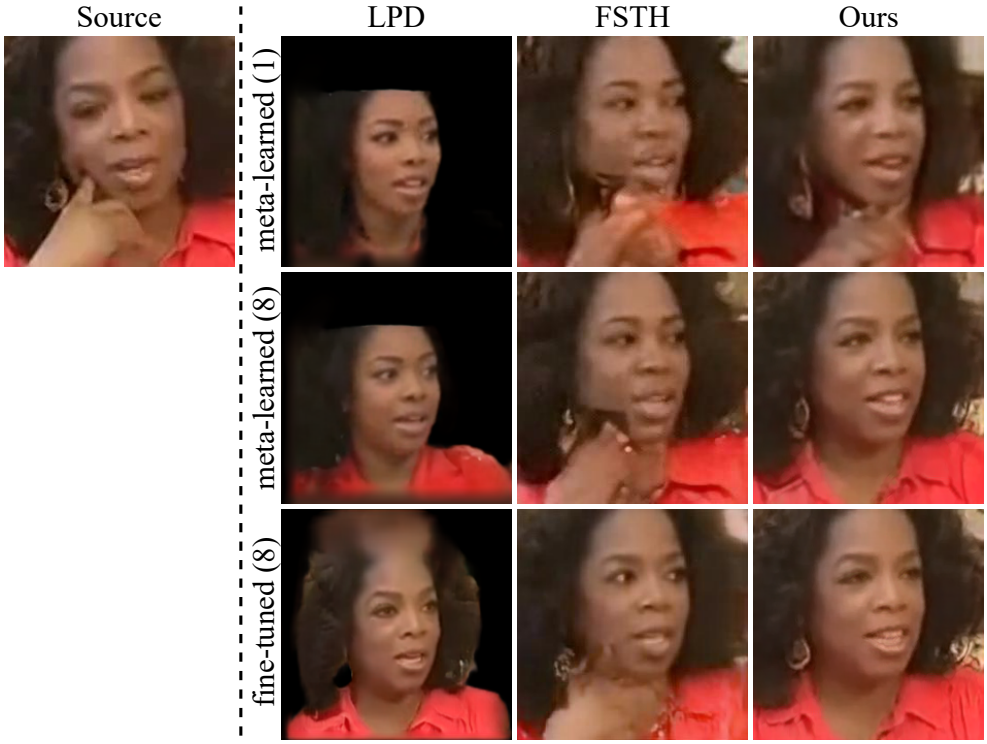


Figure 4.6: A qualitative comparison showing the effect of increasing the K-shot inputs and applying subject fine-tuning.

However, as K is increased to 32, the *fine-tuned* baselines eventually outperform our *meta-learned* model. Another very important advantage to our approach is that it achieves better performance with significantly less data. For example, fine-tuning our model with just $K = 4$ outperforms the fine-tuned baselines at $K = 32$. Since fine-tuning on more data requires more training iterations and thus more time, our method spends much less time fine-tuning on fewer data samples, and yet achieves similar or better results. We observe similar behavior with other metrics (PSNR, SSIM and LPIPS).

Figure 4.6 visualizes the effect of both increasing K and subject fine-tuning. Our method preserves the source identity without any fine-tuning, even with a single-shot input. On the other hand, the baselines only restore the source identity after the subject-specific fine-tuning. Our method also shows the most improvement, in terms of realism and better identity match, when

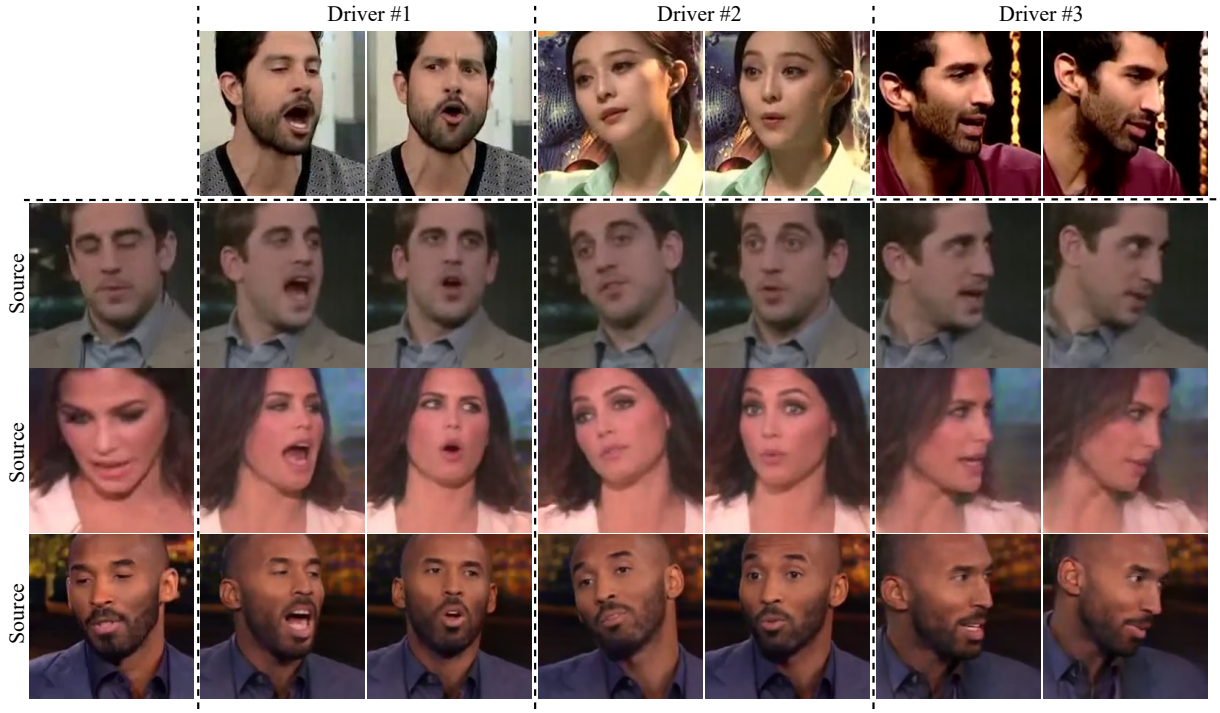


Figure 4.7: Cross-subject reenactment with different driving identities. Results are shown for our *meta-learned* model without any fine-tuning, and using 32-shot inputs.

increasing the number of K-shot inputs. For example, our method successfully filters out the subject hand occluding the face in the single-shot input.

4.4.3 Cross-subject reenactment

Cross-subject reenactment poses a challenge, especially for landmark-driven approaches. The shape difference between facial landmarks of the source and driver identities could lead to a noticeable identity gap in the reenacted results. The intermediate spatial representation learned by our method helps reduce this problem and leads to good identity preservation of the source subject regardless of the driver identity. Figure 4.7 shows sample reenactment results using different driver identities. To demonstrate the effectiveness of our disentangled representation, we avoid any subject fine-tuning and show the results of our meta-learned model with 32-shot inputs.

Table 4.2: Ablation study of our approach. +SPADE replaces the UNet generator with SPADE. +Learned seg. maps conditions the generator on learned segmentations. +Latent layout learns a latent spatial representation. The upper bound gets to cheat and uses the ground truth segmentations.

Method	PSNR↑	SSIM↑	LPIPS↓	ID-SIM↑	NMKE↓	FID↓
Baseline	17.00	0.574	0.274	0.837	0.044	67.19
+ SPADE	16.94	0.575	0.268	0.834	0.043	56.00
+ learned seg. maps	16.94	0.578	0.265	0.828	**0.042**	62.78
+ latent layout (**ours**)	**17.22**	**0.592**	**0.247**	**0.860**	0.042	**54.40**
Upper bound	18.21	0.629	0.219	0.867	0.039	48.06

The source identity is well-preserved among challenging facial expressions and different views covering both the left and right sides of the face.

4.4.4 Ablation study

We evaluate the contribution of different components of our proposed approach. All ablation experiments are trained with the same hyper-parameters and for the same number of epochs, and are evaluated in the *single-shot* setting with *no fine-tuning*. We report the results in Table 4.2. The baseline model has the same setup as FSTH [112], where a UNet generator with AdaIN layers [132] translates the input landmarks into the target image. Next, we replace the UNet architecture with a SPADE generator [133] conditioned on the facial landmarks (+SPADE). This improved the FID, but other metrics remained around the same. We hypothesize this is due to using sparse landmarks as the spatial input, while SPADE needs dense spatial inputs to generate the per-pixel denormalization parameters. To validate our hypothesis, we conducted an experiment as an *upper bound*, where we get to cheat and segment the ground truth target image using an off-the-shelf face segmentation network [137] (*i.e.* oracle), and we use these oracle segmentations as the spatial input to SPADE. Even though the oracle segmentations are noisy (*e.g.*, Figure 4.3),

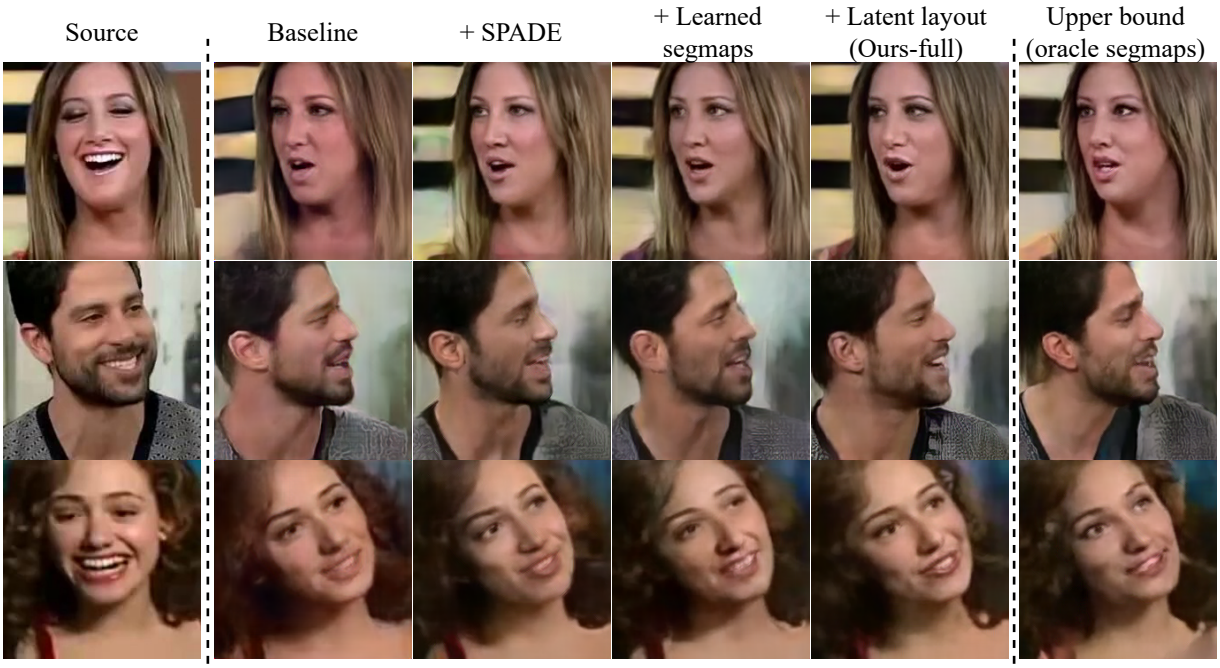


Figure 4.8: Examples from the ablation study. Results shown are for the meta-learned models with a single-shot input (source).

this still resulted in a significant boost in all metrics, proving that the SPADE generator could benefit from dense spatial inputs. Therefore, we trained a layout prediction network to predict a plausible semantic segmentation for the target pose (+Learned seg. maps). This surprisingly produced mixed results and even caused a drop in the ID-SIM and FID scores. We posit this is because the noisy oracle segmentations do not provide a consistent supervisory signal, which causes the learned segmentations to miss important shape cues (*e.g.*, the correct face shape), as well as overfit common errors in the oracle segmentations as the training progresses. Finally, removing the supervision on the predicted layouts and learning a latent spatial representations (+Latent layouts) resulted in a reasonable performance improvement over all metrics. We also show a qualitative comparison for the ablation study in Figure 4.8. We observe that the qualitative results of the upper bound experiment (using the oracle/ground-truth segmentation) exhibits artifacts caused by errors in the oracle segmentation. The results of our method with the learned

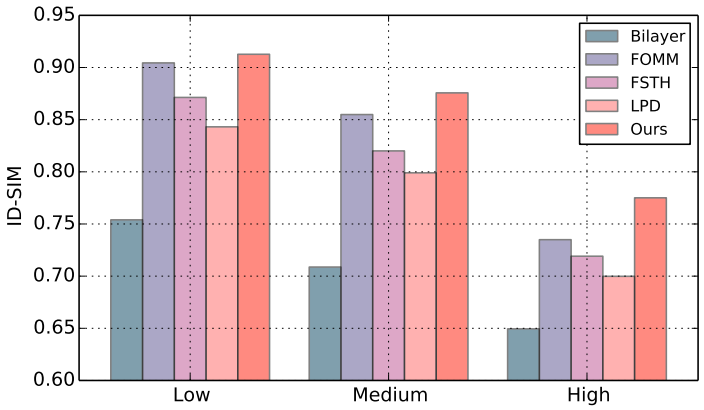


Figure 4.9: Identity similarity metric (ID-SIM) for the single-shot setting across three test subsets representing low, medium and high variance between the source and target poses. The performance gap with the baselines widens in favor of our approach as the pose variance increases.

latent layouts look qualitatively better, with no clear artifacts, despite having worse quantitative metrics than the upper bound experiment.

4.5 Additional results

4.5.1 Robustness to pose variation

Here we investigate the robustness of different methods against the pose variation between the source and the target images. First, we cluster the test set into low, medium and high pose variance based on the mean normalized keypoint difference (NMKE) between the source and target ground truth images. Then we compute the identity similarity metric (ID-SIM) per each cluster for the single-shot setting and report the results in Figure 4.9. The performance gap between our method and the baselines widens as the pose variance increases, indicating that our method has better robustness against pose variation. Note that we report the results only for the single-shot setting, where the performance gap with the FOMM baseline [3] is close. However,

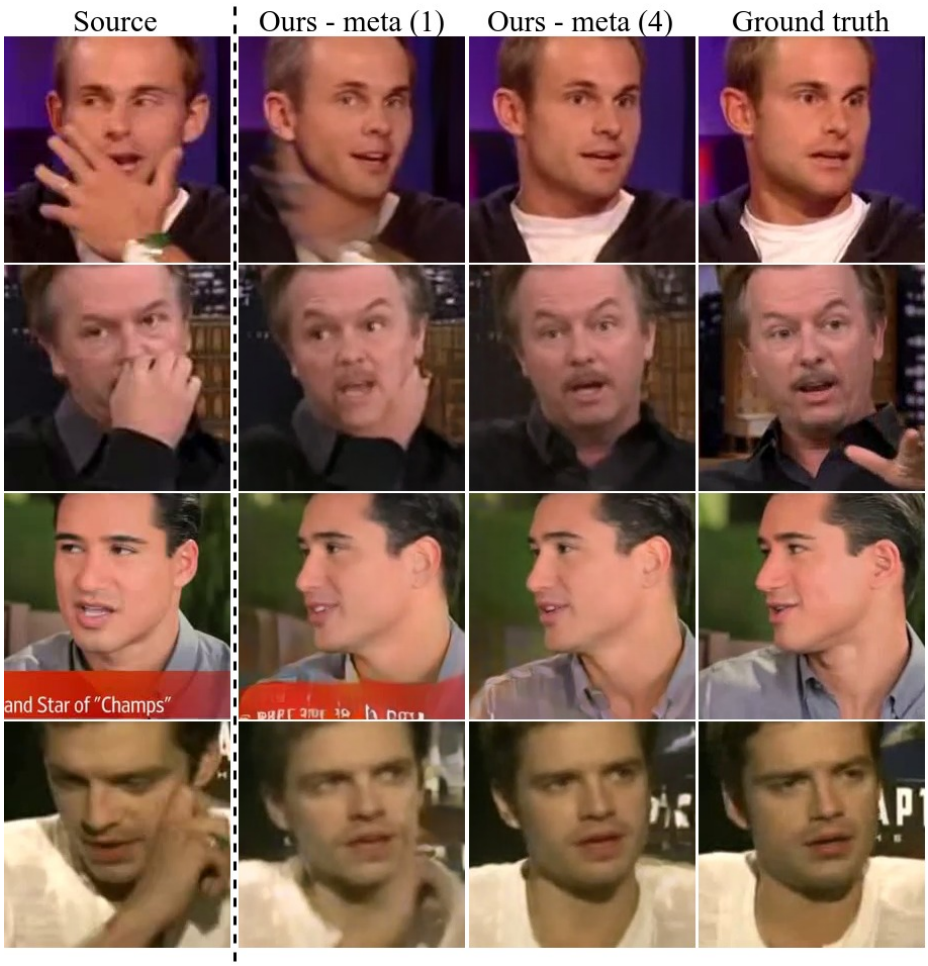


Figure 4.10: Averaging latent codes from multi-shot inputs successfully filters out transient occluders and maintains only the desired information for novel view head synthesis.

our method significantly outperforms FOMM in the multi-shot setting, as we show in § 4.5.3.1.

4.5.2 Effect of latent averaging

Given K-shot inputs, we follow [112] and obtain a single layout and style latents $\{z^l, z^s\}$ by averaging the K layout and style latents computed from the inputs respectively. We observe that averaging the K latents cancels out view-specific information and transient occluders, and successfully maintains the implicit 3D information needed for novel view head synthesis. Figure 4.10 shows some examples that highlight this effect. The single-shot source images show

Table 4.3: Comparison with the FOMM baseline [3]. While FOMM cannot benefit from multiple input frames, our method shows a significant improvement over FOMM with as few as 4-shot inputs.

Method	PSNR↑	SSIM↑	LPIPS↓	ID-SIM↑	NMKE↓	FID↓
FOMM [3]	**18.20**	**0.635**	0.236	0.869	0.061	56.10
Ours-meta (K=1)	17.27	0.598	0.241	0.869	0.041	48.11
Ours-ft (K=1)	17.37	0.605	**0.232**	**0.886**	**0.041**	**45.69**
Ours-meta (K=4)	18.90	0.638	0.192	0.909	0.039	43.19
Ours-ft (K=4)	**19.33**	**0.661**	**0.171**	**0.930**	**0.037**	**34.31**

some transient occluders like the subjects' hand or news bar, which in turn corrupts our single-shot output. However, increasing the inputs to four frames successfully filters out the transient occluders and results in clean outputs.

4.5.3 More comparative results

4.5.3.1 Comparison with FOMM

The FOMM baseline [3] accurately reconstructs the background and other static regions due to its warping-based nature. Therefore, it achieves lower reconstruction error (PSNR and SSIM) than our approach in the single-shot setting, even if their output contains clear artifacts in the face area. However, one limitation to FOMM is that it cannot utilize more input frames to its advantage. On the other hand, Table 4.3 shows that our approach benefits from as few as four input frames to outperform FOMM, even in the meta-learned mode. Subject fine-tuning further improves our performance to outperform FOMM by a wide margin on all metrics.

Table 4.4: Detailed quantitative comparison with the few-shot baselines, showing the effect of both increasing the K-shot inputs and subject-specific fine-tuning.

K	Method	No Subject Fine-tuning						Subject Fine-tuned					
		PSNR↑	SSIM↑	LPIPS↓	ID-SIM↑	NMKE↓	FID↓	PSNR↑	SSIM↑	LPIPS↓	ID-SIM↑	NMKE↓	FID↓
1	FSTH	16.80	0.570	0.259	0.801	0.048	51.12	16.92	0.597	0.263	0.836	0.049	53.07
	LPD	–	–	–	0.732	0.072	80.20	–	–	–	0.837	0.070	48.48
	Ours	**17.27**	**0.598**	**0.241**	**0.869**	**0.041**	**48.11**	**17.37**	**0.605**	**0.232**	**0.886**	**0.041**	**45.69**
4	FSTH	–	–	–	–	–	–	–	–	–	–	–	–
	LPD	–	–	–	0.755	0.069	79.67	–	–	–	0.909	0.058	38.81
	Ours	**18.90**	**0.638**	**0.192**	**0.909**	**0.039**	**43.19**	**19.33**	**0.661**	**0.171**	**0.930**	**0.037**	**34.31**
8	FSTH	17.86	0.600	0.225	0.836	0.046	46.38	18.35	0.647	0.218	0.899	0.044	45.15
	LPD	–	–	–	0.760	0.068	77.00	–	–	–	0.922	0.056	35.87
	Ours	**19.18**	**0.645**	**0.186**	**0.917**	**0.039**	**44.23**	**19.65**	**0.675**	**0.160**	**0.940**	**0.036**	**32.54**
32	FSTH	18.66	0.613	0.207	0.843	0.044	44.85	19.69	0.686	0.171	0.927	0.041	33.69
	LPD	–	–	–	0.769	0.066	63.47	–	–	–	0.935	0.054	33.96
	Ours	**19.35**	**0.650**	**0.182**	**0.921**	**0.037**	**43.32**	**19.98**	**0.690**	**0.146**	**0.948**	**0.038**	**28.26**

Figure 4.11: Extending Figure 4.6. More results comparing our performance to the few-shot baselines with respect to increasing the the K-shot inputs and applying subject fine-tuning.

4.5.3.2 More comparative evaluation

We report the quantitative details for the effect of increasing the number of K-shot inputs, as well as the effect of subject fine-tuning in Table 4.4. We observe similar conclusions to those obtained from Figure 4.5. LPD [113] performs very poorly in the meta-learned setting, and only outperforms the FSTH baseline [112] in the subject fine-tuning setting. On the other hand, our method consistently outperforms the baselines on all metrics across different settings. Further-

more, the performance of our method at $K = 4$ is on-par with or outperforms the baselines evaluated at $K = 32$ across all metrics. Since the LPD [113] baseline does not predict the background and re-crops the input/output frames, we subtract the background and compare with their corresponding cropped ground truths for quantitative analysis. We also exclude LPD from frame reconstruction evaluation since its outputs do not align with the rest of the methods. Also, the authors of FSTH [112] only provide their output for $K = \{1, 8, 32\}$ and they did not release their code. Therefore, we don't report their performance for $K = 4$.

4.5.3.3 More qualitative comparisons

Here, we expand our qualitative comparisons of §§ 4.4.1 and 4.4.2 in both the multi-shot and single-shot settings. Figure 4.11 extends Figure 4.6 and shows more examples comparing the effect of increasing the K-shot inputs and applying subject fine-tuning between our method and the baselines. Figure 4.12 shows more comparisons in the single-shot setting as Figure 4.4.

4.5.4 More qualitative results

We show more qualitative results of our method showing the effect of increasing the K-shot inputs, and the effect of applying the subject fine-tuning in Figure 4.13. We observe that we get a noticeable improvement when we increase K from 1 to 4. The visual gain from increasing K further starts to saturate, although quantitative metrics generally keep improving (*e.g.*, Table 4.4). While increasing K beyond 4 still leads to better visual results in general, we observe that the most improvement focuses on the background and clothes reconstruction, with slight improvements to sharpness and color matching as well. Subject fine-tuning further improves the sharpness and

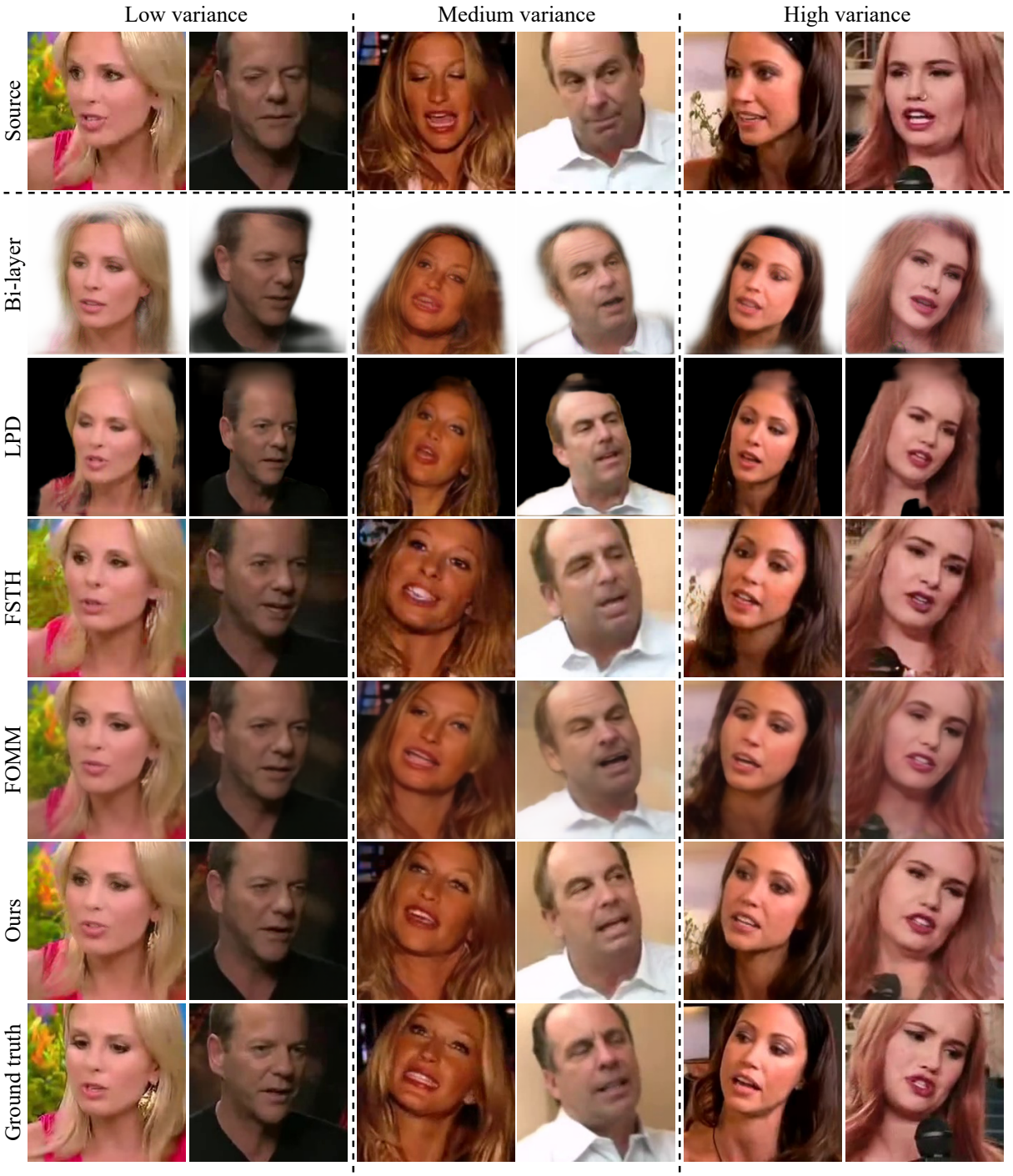


Figure 4.12: Extending Figure 4.4, showing more qualitative comparisons in the single-shot setting. We show three sets of examples representing low, medium and high variance between the source and target poses. Our method is more robust to pose variations than the baselines.

better reconstructs the background details.

Figure 4.13: Qualitative results of our method showing the gains of increasing the number of K-shot inputs and applying subject fine-tuning.

Figure 4.14: Expanding Figure 4.7 by showing the same reenactment results in the single and 4-shot settings. Our model extrapolates well to challenging poses and expressions even with a single-shot input (shown in source), while preserving the source identity.

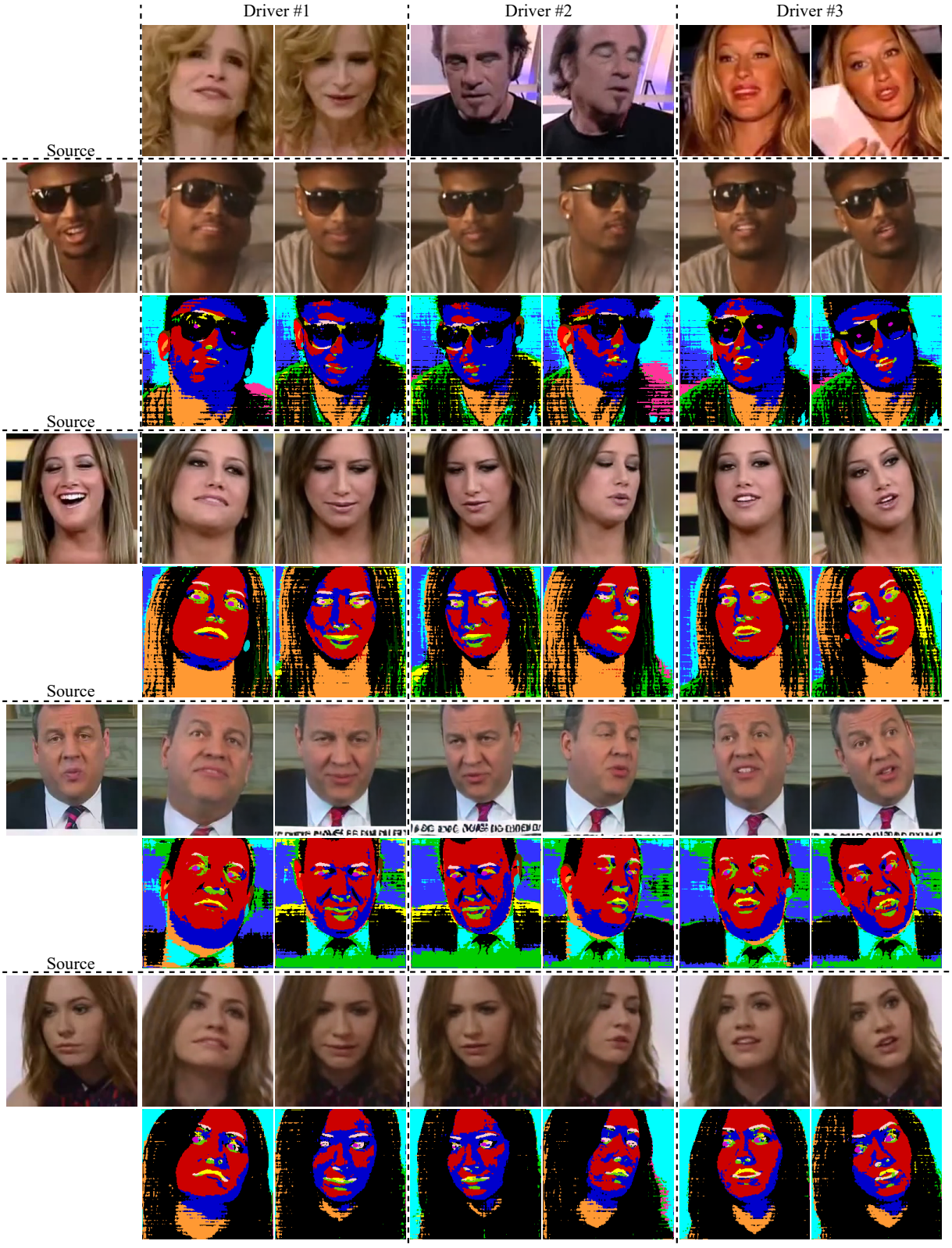


Figure 4.15: More cross-subject reenactment results with different driving identities. Results are shown for our *meta-learned* model without any fine-tuning, and using 32-shot inputs. We also show the corresponding latent spatial layout maps.

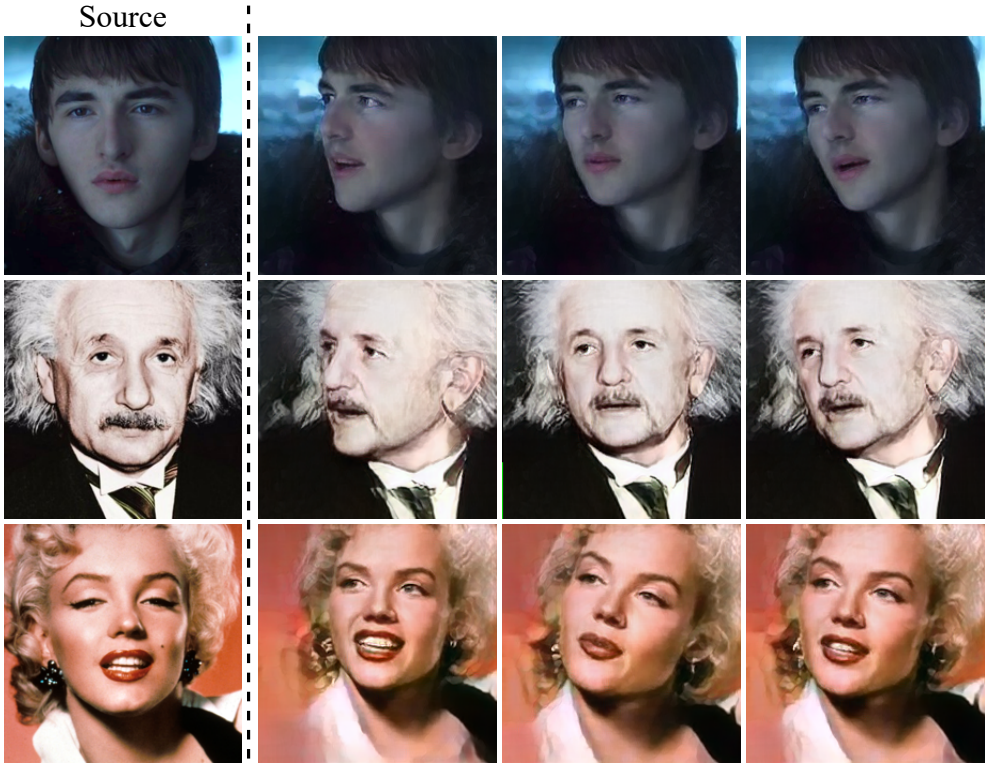


Figure 4.16: Qualitative results on subjects not belonging to VoxCeleb2.

4.5.5 More reenactment results

We first expand Figure 4.7 by showing the same reenactment results but for the single-shot meta-learned setting and the 4-shot inputs in both the meta-learned and subject fine-tuned settings in Figure 4.14. The results show that even in the single-shot meta-learned setting, our model does a pretty good job extrapolating the input image (source) to challenging poses and expressions, while preserving the source identity. Increasing the input shots to 4 leads to a noticeable visual improvement, and fine-tuning further leads to slight improvements, most notable in the female source (middle example). These results show that our method does not require many frames to produce realistic and identity preserving results. For video comparison with the baselines, please refer to the supplementary videos[1].

We also extend Figure 4.7 by showing more reenactment results in Figure 4.15. We also show the predicted spatial layouts corresponding to the outputs. The predicted spatial layouts may be less interpretable than traditional semantic segmentations, but they seem to encode more information and capture accurate details about the face shape.

Additionally, we perform out-of-domain reenactment using source subjects not present in the VoxCeleb2 dataset. Some qualitative results are shown in Figure 4.16. Our approach can synthesize realistic novel views given only a single-shot input, although in some cases it shows a bit of an identity gap.

4.5.6 Limitations and failure cases

Here we discuss some of the limitations of our approach.

[1]Supplementary videos can be found at `http://www.cs.umd.edu/~mmeshry/projects/lsr/`

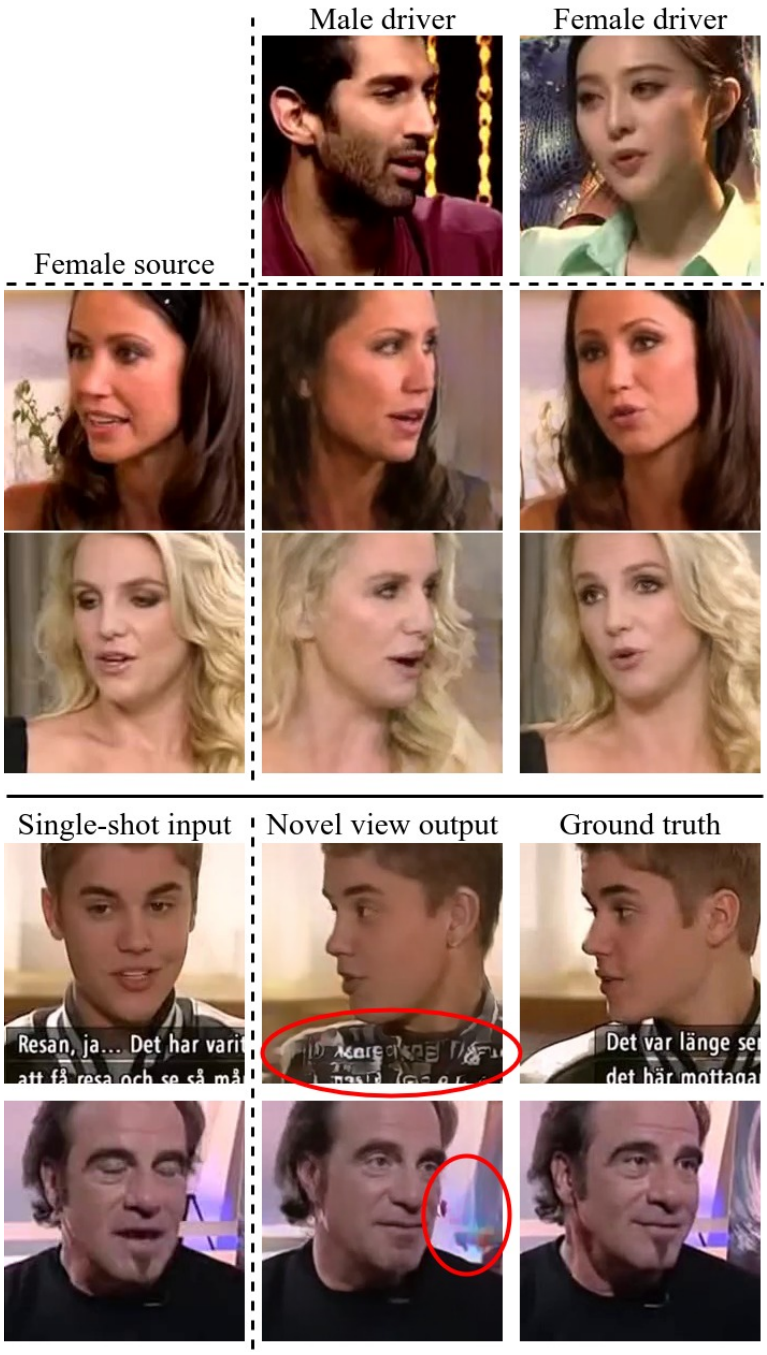


Figure 4.17: Example limitations. Top: male-to-female reenactment sometimes causes low identity preservation and other visible artifacts. Bottom: our approach cannot faithfully reconstruct the background details.

- **Temporal consistency.** Similar to previous direct synthesis approaches [112, 113, 114], our training does not enforce temporal consistency. Therefore the output videos often

contain some flickering. Considering the temporal aspect during training (*e.g.*, similar to [127]) could mitigate this problem, but at the expense of higher training cost.

- **Failure modes for cross-subject reenactment.** We observe that most of the failure cases in cross-subject reenactment are caused by either source subjects with complex backgrounds, or using male drivers to animate female sources (*e.g.*, Figure 4.17). Since complex backgrounds could lead to artifacts in our results, then this signifies that the background information is entangled with the face and identity information. Learning a better disentangled representation could improve this problem. On the other hand, the trouble faced with male-to-female reenactment implies that our approach still has some sensitivity to the driver landmarks. While our approach reduces this sensitivity significantly compared to previous baselines, there is still room for improvement.

- **Background reconstruction** Direct synthesis approaches, including our method, synthesize the target frame from a compressed latent code. This compressed bottleneck leads to the loss of some information, especially for the background details. Figure 4.17 shows some examples in the single-shot setting. Our method cannot transfer static parts (*e.g.*, the closed captions or the background) from the source image to the synthesized view. Borrowing elements from the warping-based approaches is one direction to better reconstruct static details.

- **Dataset-induced limitations.** The VoxCeleb2 dataset [139] has low resolution videos and is processed to perform zoomed-in center crops that often cut off the top of the head. Dataset biases are inevitably inherited by the trained models. Therefore, generating output for out-of-domain inputs requires pre-processing the inputs to have similar properties to

the VoxCeleb2 dataset.

4.5.7 Ethical concerns

While the task of synthesizing realistic *talking heads* has a wide range of applications, it also raises ethical concerns regarding potential misuses of this technology. A prime example of this is the growing misuse of DeepFakes [143, 144]. Several state-of-the-art methods can easily swap identities, expressions as well as face attributes and generate photo-realistic samples. Additionally, with the increase in the ease of access to face reenactment models, more and more people can misuse such models through widely available applications. Thus, it is important at the same time to have the ability to detect fake content. In this direction recent works like [145, 146, 147] have tried to solve the problem of detecting real *vs.* fake images. Especially interesting is the work by Wang *et al.* [146] which shows that models for fake image detection can be made to generalize well to unseen scenarios. While this is a temporary respite, it is important to continue research in the field of fake image detection to keep on par with the ever improving field of image synthesis, as not only do models improve, but also the ease of access to such models grows rapidly.

4.6 Implementation details

Dataset pre-processing. The released VoxCeleb2 dataset [139] contains pre-processed videos to have a center crop around the face. We uniformly sample 10 frames from each video and obtain the facial landmarks using an off-the-shelf facial landmarks detector [140]. Once the landmarks

are obtained we use the same procedure as [112] to connect the facial landmarks to obtain contours for different face parts (*e.g.*, eyes, nose, lips ...etc). We observe that the facial landmarks extraction fails for a small fraction of videos, which we opted to ignore. We also segment each frame using the face parsing tool provided by [137] to obtain the oracle segmentation maps for pre-training the layout prediction network. The face parsing network performs poorly on VoxCeleb2 frames due to the domain gap, in terms of image resolution and the distribution of head poses, between the datasets used to train the face parsing network [137], and the cropped VoxCeleb2 videos. We observe that the face segmentation network [137] better captures different details at different resolutions. For example the segmentation result at the original VoxCeleb2 resolution better captures larger regions like the hair, neck and clothes. On the other hand, upsampling the frame to the resolution used for training the segmentation network [137] gives better segmentation results for the finer and smaller regions like the nose, eyes, mouth, and ears. So, to improve the oracle segmentations, we segment each frame twice at 256×256 and 512×512 resolutions and merge the coarse and fine semantic classes from both results.

Encoder networks. We use a resnet encoder for both the layout and style encoders $\{E^l, E^s\}$. The encoder architecture has 5 downsampling blocks, followed by a fully connected layer that generates a 512-dimensional latent code. The architecture for the residual blocks is borrowed from [12], with replacing *average-pooling* with *blur-pooling*. We use 32 feature maps at the first encoder layer and double this number after each downsampling block with a maximum of 512 feature maps. We follow [112] and concatenate the facial landmarks to the few-shot RGB images before feeding them to the encoder.

Layout generator. We use a traditional UNet architecture [136] with residual blocks. The residual blocks are borrowed from [12] with replacing *BatchNorm* with *InstanceNorm* and applying adaptive instance normalization (*AdaIN*) [132]. The smallest and largest number of feature maps are 32 and 512 respectively, and we use *blur-pool* and *bilinear* upsampling in the downsampling and upsampling blocks respectively.

Image generator. We use a SPADE generator architecture [133] with replacing *BatchNorm* with *InstanceNorm*. We also use 32 feature maps at the last generator layer and 64 feature maps in each SPADE block, compared to 64 and 128 feature maps respectively in the original SPADE architecture [133]. The input to each SPADE block is the concatenation of the predicted layout map and the facial landmarks.

Discriminator. We borrow the architecture of the discriminator network from [14], with reducing the smallest number of feature maps from 64 to 32. We also use a non-saturating logistic loss with gradient penalty [138].

Training. We follow [11] and use equalized learning rate in all of our networks. We pre-train the layout prediction network for 2 epochs, followed by training the full pipeline for 8 epochs. Our best model was left to train for an extra 5 epochs, which mainly improves the FID score, while slightly improving the other quantitative metrics as well. We use an *Adam* optimizer [148] with $\beta_1 = 0, \beta_2 = 0.999$, and a learning rate of 0.001 for all networks. We linearly decay the learning rate by a factor of 100 during the last epoch.

4.7 Conclusion

In this chapter, we presented *LSR*, a framework for few-shot talking-head synthesis. Given as few as a single shot of a target subject, our model learns a novel latent spatial representation that proves effective for rendering the target subject under novel head poses and facial expressions. LSR improves the performance of both subject-agnostic models, as well as subject-finetuned models while requiring significantly less data samples. The learned latent spatial representation helps provide robustness against a wide range of poses and expressions, and results in better identity preservation, especially for the cross-subject reenactment scenarios.

Chapter 5: RTMV: A Ray-Traced Multi-View Synthetic Dataset for Novel View Synthesis

Recent implicit neural representations have had a revolutionary impact on novel view synthesis research. Neural Radiance Fields [40] (NeRF) in particular has incited an explosion of follow-ups and extensions that pushed novel view synthesis to new frontiers. However, with this explosion of research works, there is a pressing need for a unified benchmark to evaluate these works and challenge them to uncover their limitations. In this chapter, we take a step back and address novel view synthesis in its standard setting, where we have a static scene with many observations of the scene. We first present a large-scale synthetic dataset for novel view synthesis consisting of $\sim$300k images rendered from nearly 2000 complex scenes using high-quality ray tracing at high resolution (1600×1600 pixels). The dataset is orders of magnitude larger than existing synthetic datasets for novel view synthesis, thus providing a large unified benchmark for both training and evaluation. Using 4 distinct sources of high-quality 3D meshes, the scenes of our dataset were composed to exhibit challenging variations in camera views, lighting, shape, materials, and textures. Because our dataset is too large for existing methods to process, we propose Sparse Voxel Light Field (SVLF), an efficient voxel-based light field approach for novel view synthesis that achieves comparable performance to NeRF [40] on synthetic data, while being an order of magnitude faster to train and two orders of magnitude faster to render. SVLF

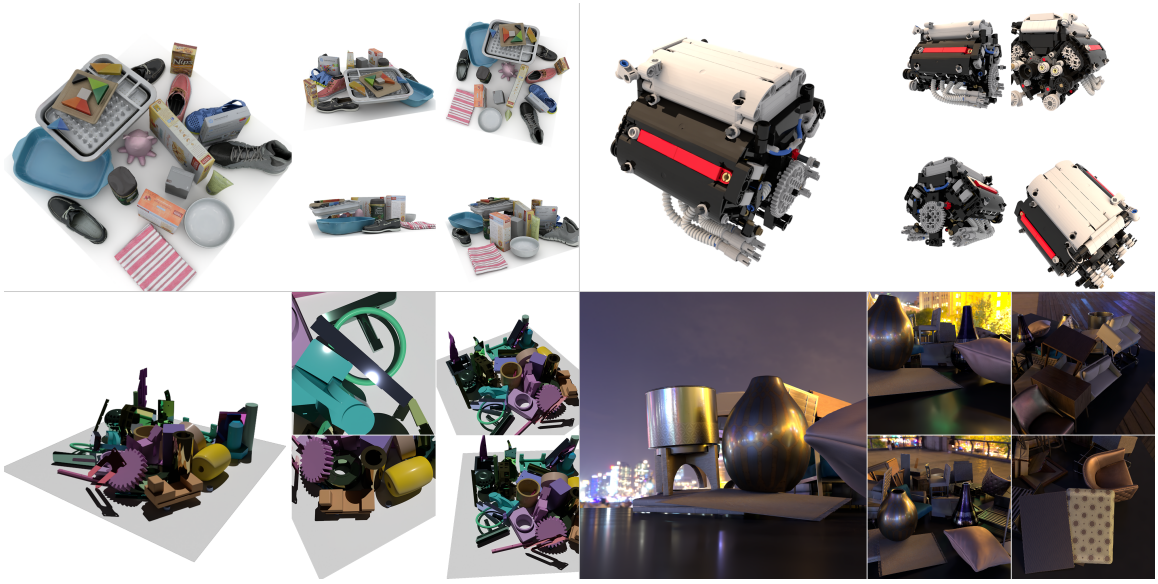


Figure 5.1: We present RTMV, a large-scale high-fidelity ray-traced synthetic dataset for novel view synthesis. RTMV is composed of nearly 2000 scenes from 4 different environments exhibiting large varieties in view positions, lighting, object shapes, materials, and textures. Each quadrant shows a single scene from each environment, captured from multiple viewpoints.

achieves this speed by relying on a sparse voxel octree, careful voxel sampling (requiring only a handful of queries per ray), and reduced network structure; as well as ground truth depth maps at training time. In contrast to global implicit neural representations, such as NeRF, SVLF proposes a hybrid representation that combines an explicit geometric proxy (*e.g.*, a coarse voxel grid) and local implicit representations to render high quality results.

5.1 Introduction

Since the publication of NeRF [40], there has been an explosion of interest in neural volume rendering for novel view synthesis [9, 149, 150, 151, 152, 153, 154]. This heightened attention of the research community is due to the impressive high-fidelity results that had long remained elusive but which are now possible. Further, these techniques are able to capture the 3D geometry of the scene with precision [155, 156, 157], thus opening up an entirely new family of algorithms

for 3D scene reconstruction.

Despite this progress, it is not yet clear what the limitations of such techniques. are Because of the difficulty of acquiring multi-view images of complex scenes with ground truth, only a handful of scenes have been available to date. As a result, since the existing methods have only been tested on a small number of scenes, it is unclear how they will perform on a wider variety of scenes. For example, can these methods simultaneously handle multiple objects, diverse materials and textures, challenging lighting conditions, and free camera poses? Can they generalize to new complex scenes from a few views? Questions such as these point to the need for a large-scale dataset to evaluate algorithms for novel view synthesis.

In this work, we propose to address this problem by sharing a large-scale dataset for novel view synthesis. Comparison to existing datasets is shown in Table 5.1. Our dataset consists of nearly 2000 scenes composed of multiple objects, with complex materials and textures illuminated by various light sources (see Figure 5.1). We selected these objects from 4 distinct sources of large collections of 3D meshes, thus providing significant variety to the dataset. Approximately 300k high-resolution (1600×1600) synthetic images were created using ray tracing to ensure high fidelity, with the virtual camera placed either on a hemisphere or freely within the environment (see Figure 5.2). Using synthetic data enables the following advantages: 1) noise-free annotations, 2) rich metadata that otherwise would not be possible, and 3) full control over variations. Moreover, recent efforts at real-time ray tracing have made tremendous progress in reducing the sim-to-real gap for different computer vision applications, *e.g.*, pose estimation [158] or facial analysis [159]. As a result, synthetic data is becoming increasingly important for training and evaluating deep networks [160, 161, 162].

We use this dataset to benchmark several existing algorithms, thus highlighting existing

limitations and pointing out the need for future research. However, we are only able to evaluate available methods on a small subset of our dataset due to their high computational demands. To reduce the computational cost, we propose a novel algorithm called SVLF (Sparse Voxel Light Field) that is an order of magnitude faster than NeRF to train on synthetic scenes, and nearly two orders of magnitude faster to render. This speedup is achieved by a novel combination of an octree representation, querying each voxel only once, and using a much smaller network, as well as leveraging depth maps for training. We show that SVLF achieves results comparable to NeRF while being much more computationally efficient.

Our dataset has many uses beyond evaluation. First, a large dataset like ours is useful for training neural networks by exposing the network to more variety than is possible with existing datasets. Second, the size of the dataset allows for investigating few-show view synthesis, where an algorithm trained on a larger set of scenes is then able to process new scenes without training from scratch. Third, because the synthetic data allows for rich metadata such as ground truth depth, geometry, camera poses, object positions, and so forth, it facilitates research on networks that require these inputs for training (*e.g.*, novel view synthesis algorithms that train on ground truth depth). Finally, the dataset can be used for problems other than novel view synthesis, such as 3D reconstruction and pose estimation.

5.2 Related work

Novel view synthesis. Novel view synthesis is a long-standing problem in computer vision and graphics. Image-based rendering (IBR) techniques [92, 93, 168, 169, 170, 171, 172, 173] synthesize a novel view from a set of reference images by computing blend weights of nearby

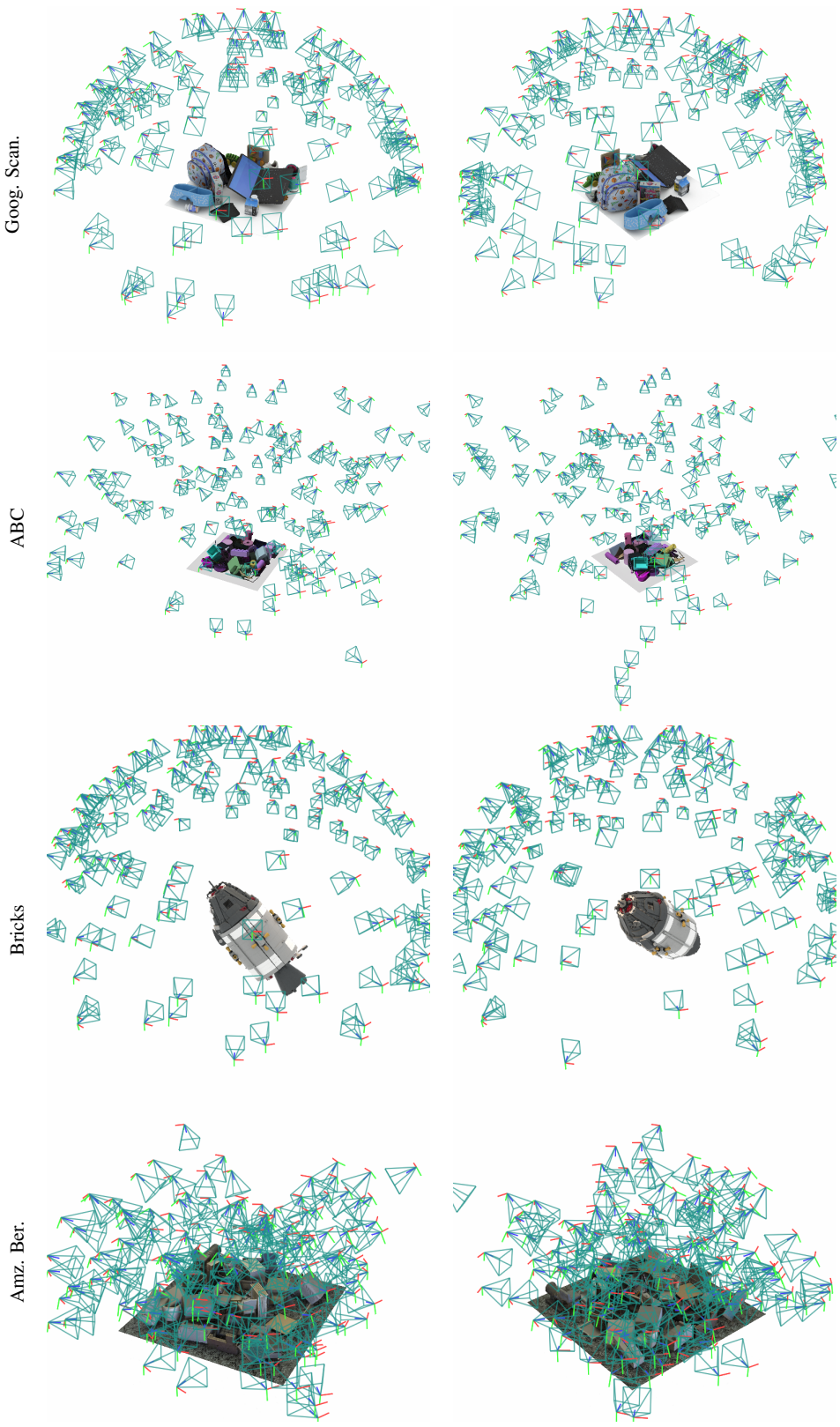


Figure 5.2: Camera distributions for a single scene per environment, for each scene we present 2 example view points.

Table 5.1: Comparison of novel view synthesis datasets. From top to bottom: the number of scenes, image resolution (smallest dimension), whether the scene is composed of multiple objects, whether the camera can freely move about the scene (as opposed to front-facing, hemisphere, etc.), whether ground truth (GT) information is available (*e.g.*, depth, lighting, object material), and whether high dynamic range (HDR) is exported.

Datasets	NeRF [40]	T&T [163]	B-MVS [164]	DTU [165]	SRN [9, 166]	LLFF [6]	CO3D [167]	**Ours**
scenes	8	15	502	124	3511	24	20k	2000
resolution	800	1080	1536	1200	128	480	1088	1600
multi-object	✗	✗	✓	✓	✗	✗	✗	✓
free camera	✗	✓	✓	✗	✗	✗	✗	✓
full GT	✓	✗	✗	✗	✓	✗	✗	✓
HDR export	✗	✗	✗	✗	✗	✗	✗	✓

reference views. Another class of work builds a volumetric representation of the scene such as voxel grids [172, 174, 175, 176, 177] or multi-plane images (MPI) [6, 178, 179, 180, 181, 182]. To render such representations, techniques like alpha compositing or volume rendering are typically used to synthesize novel views.

Implicit neural representations. More recently, implicit neural representations [37, 38, 39] have shown great potential for novel view synthesis by storing scene properties as the weights of a fully connected network. Most notably, NeRF [40] optimizes a 5D radiance field that maps a point and viewing direction to its density and color values, which can then be integrated for rendering. Mip-NeRF [149] uses integrated positional encoding over a cone frustum to render anti-aliased multi-scale images. NeRF has opened the door to many exciting research directions [149, 150, 153, 154, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194]. However, NeRF (and closely-related variants) are slow to train and render, as rendering a single pixel requires evaluating hundreds of samples along a ray, each constituting an expensive neural network query. To increase speed, some works [151, 195] utilize an octree to prune empty space and only sample points close to the

surface. Another alternative, DONeRF [152], utilizes depth images to train a depth prediction network, thus achieving high-quality synthesis by evaluating only a few point samples around the estimated depth. Instant-NGP [4], which was developed concurrently with this work, is yet another alternative that uses multiple tricks to achieve orders of magnitude speedup over NeRF.

While NeRF focuses on fitting single scenes, several works aim to generalize across multiple scenes [9, 196, 197, 198]. First, a prior is learned over a feature volume by training on a dataset of multiple scenes. Then, presented only with few-shot images of a new scene, the learned prior is utilized to render novel views of the new scene.

Data synthesizer. Given the high demand for large annotated datasets for deep learning, there has been an increase in both synthetic datasets [199, 200, 201, 202, 203, 204, 205] and in tools for generating such data [206, 207, 208, 209, 210]. The Cycles renderer included in Blender has been widely used in the research community for generating synthetic data because of its ray tracing ability [40, 211, 212, 213, 214, 215, 216]. In an attempt to more easily generate synthetic images, Denninger *et al.* [209] introduced an extension to Blender that renders objects falling onto a plane with randomized camera poses. Other tools, like SAPIEN [210], have also used the OptiX backend [217] to optimize ray tracing performance. NViSII [218] adds to this suite by providing scripting capabilities, along with the ease of installation and development.

Datasets for novel view synthesis. The original NeRF paper [40] introduced 8 synthetic scenes with 360-degree views. Since their release, the accessibility of these scenes has allowed for rapid progress in novel view synthesis, yet, the small number of such scenes makes it difficult to assess performance at scale. Wang *et al.* [197] introduced a synthetic dataset including Google Scanned Objects. Other datasets for training multi-view algorithms include DTU [165], LLFF [6], Tanks

and Temples [163], Spaces [181], RealEstate10K [180], SRN [9, 166], Transparent Objects [219], ROBI [220], CO3D [167, 221], SAPIEN [210], and BlendedMVS [164]. See Table 5.1 for further information. During the preparation of this manuscript, Reizenstein *et al.* [167, 221] introduced a large real dataset for multi-view synthesis, using camera views around the object. We believe our proposed dataset nevertheless provides unprecedented scale, variety, and quality.

5.3 Ray-Traced Multi-View Dataset

5.3.1 NViSII

Recent advances in ray tracing have allowed researchers to generate photo-realistic images with unprecedented speed and quality. Such advancements are accessible via tools integrated within existing rendering frameworks with complex interfaces (*e.g.*, [209]). In contrast, we propose an accelerated standalone Python-enabled ray tracing / path tracing renderer built on NVIDIA OptiX [217] with a C++/CUDA backend. The Python API allows for non-graphics experts to easily install the renderer, quickly create 3D scenes with the full power that scripting provides, and is simply installed through `pip` package management.

NViSII allows for complex visual material definitions using Disney's principled BSDF [222] model. This model takes sixteen parameters, whose combinations result in physically-plausible, well-behaved results across their entire range. These parameters, which include color, metallic, transmission, and roughness components, have a perceptually linear effect on the appearance of the BSDF.The linearity property is well-suited to uniform random sampling in order to create a variety of different materials, such as smooth and rough plastics, metals, and dielectrics (*e.g.*, glass and water). We drive these parameters using either scalar values or textures to create a

Table 5.2: High-level parameters for the four environments.

Environment	Camera type	Lighting	# scenes
Google Scanned	hemisphere	white dome	300
ABC	free	spotlight	300
Bricks	hemisphere	white dome & sun	1027
Amazon-Berkeley	free	HDRI dome light	300

diverse dataset.

Our solution also supports advanced rendering capabilities, such as multi-GPU ray tracing, physically-based light controls, accurate camera models (including defocus blur and motion blur), native headless rendering, and exporting various metadata (*e.g.*, segmentation, motion vectors, optical flow, depth, surface normals, and albedo).

5.3.2 Data generation

We use NViSII [218] to generate 1927 unique scenes in total from 4 different environments, each environment being a collection of 3D models drawn from a different data source (see Table 5.2). For each scene, we normalize it to be contained within a unit cube and generate 150 unique image frames. Ground truth annotations include a depth/distance map, background alpha mask, segmentation mask, scene point cloud, and various metadata (*e.g.*, camera poses, entity poses, material properties, and light positions). Images are exported in the 'EXR' format as it allows for storing colors in a high dynamic range without tonemapping and/or gamma correction pre-applied. Each image is rendered at 4000 samples per pixel. We use 2 types of camera placements: hemisphere and free (*e.g.*, Figure 5.2). The former places the camera on the hemisphere surrounding the scene, while the latter moves the camera freely within a cube avoiding collisions with the objects. Overall the dataset is orders of magnitude larger than what is typically used

for novel view synthesis. Since it is time consuming for an algorithm like NeRF [40] to process the entire dataset, we also provide a smaller subset composed of 40 scenes, 10 from each environment.

5.3.3 Environments

These 4 different environments are as follows.

Google Scanned Objects. All the ∼1000 Google scanned models [223] were used to generate 300 falling objects scenes. In each scene, we randomly selected 20 3D models, added collision shapes, and let the objects fall onto each other and onto a table using the PyBullet physics simulator [224]. We applied a random material to the texture since BRDF materials are not provided with the models (they are mostly plastic-like). Note that the provided textures from these 3D models have light baked into the textures, which can cause the object to look unrealistic, *e.g.*, having highlights where it should be dark. The camera remains at a constant distance from the center and always points toward the center of the scene. The scenes are lit by a single white dome light. This dataset is the most similar to NeRF's synthetic data [40]. This is also the easiest of our 4 environments: it has a small number of objects and views distributed on the hemisphere.

ABC. We used all the 3D models (∼1.5M) from the ABC dataset [225]. In each scene, we randomly selected 50 3D models and scaled them so that they are all of a similar size. Similar to the previous environment, we constructed 300 scenes of falling objects using physical simulation. Each object was also given a random BRDF material (plastic, metallic, rough metallic, *etc.*), and a random bright color, so that this environment has multiple view-dependent materials. The scenes were illuminated by a uniform dome light, and by a bright point light simulating a directional

light to produce hard shadows. The camera was allowed to freely move within the unit cube and look at any of the objects, thus producing both close and far images. As a result, this environment has the most challenging viewpoints.

Bricks. We downloaded 1027 unique models from Mecabricks. For each model, we generated a single scene where the model was placed in the middle of the scene. The camera was randomly placed on a hemisphere at a fixed distance from the center. The camera was aimed at random locations within 1/10 of the unit volume used to scale the object, thus producing images that are not centered on the model. Each scene was illuminated by a white dome light and a warm sun placed randomly on the horizon. These detailed scenes are challenging due to the variety of textures and presence of self-reflections. This is our largest environment, since it has 1027 scenes.

Amazon-Berkeley. Similar to ABC and Google Scanned Objects, we loaded 40 3D models from the Amazon Berkeley Objects (ABO) dataset [226], scaled them, and let them fall onto a plane. Since the 3D models are accompanied with high-quality textures and material definitions, we light the scene with a full HDRI map and apply a random texture on the floor, both from Poly Haven. Similar to ABC scenes, the camera was allowed to move freely within the unit cube and to look at any object. Also similar to ABC, the materials are quite challenging, such as metallic (mirror-like) objects. We believe this is the most challenging environment of the dataset offering interesting views, materials, textures, lighting, and photo-realistic backgrounds.

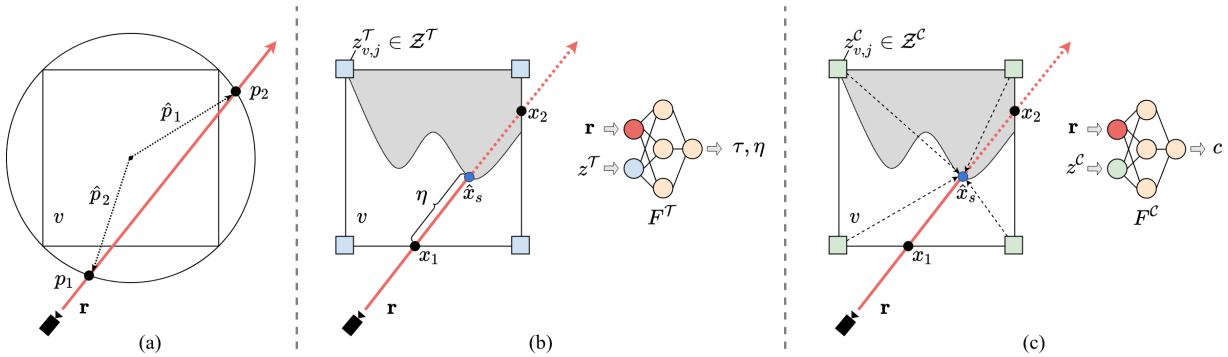


Figure 5.3: SVLF overview for a given voxel v. (a) Rays are parameterized by the normalized intersection points with the bounding sphere, $\mathbf{r} = [\tilde{p}_1 \mid \tilde{p}_2]$, where $\tilde{p}_i = \hat{p}_i / \|\hat{p}_i\|$. (b) SVLF first estimates the optical thickness τ and a within-voxel depth η to the surface hit, if any. (c) Color features are sampled at the estimated surface point $\hat{x}_s$ and a color decoder predicts the final ray color c.

5.4 Sparse Voxel Light Field (SVLF)

Given the size and complexity of our dataset, in this section we present an efficient and lightweight multi-view voxel-based algorithm for novel view synthesis. At a high level, our method learns a voxel-based function that maps a ray $\mathbf{r}$ to the optical thickness τ_{v_i}[1] and color c_{v_i} associated with a particular voxel v_i. This mapping is realized by evaluating two small decoder networks. Rendering a single pixel consists of a volumetric integration that amounts to only a handful of network queries, determined by the number of voxels intersected along the ray. This is in contrast to evaluating hundreds of queries along the ray as in NeRF [40].

5.4.1 Representation

Scene volume. Given depth maps provided with our dataset, we partition the scene into a voxel

[1]Optical thickness is defined, as in [227], as the integral of the density along a segment of a ray, so that $\alpha = (1 - e^{-\tau_{v_i}})$ is the opacity.

grid $\mathcal{V}$ and represent the space using an octree [157]. This allows for efficient memory utilization as the octree only represents voxels that contain a surface. We use $\mathcal{Z}$ to represent a collection of feature vectors over the feature volume. Each voxel v in the octree stores learnable feature embeddings $z_{v,j} \in \mathcal{Z}$ at each of its eight corners ($j \in \{1, 2, \dots 8\}$), where feature embeddings are shared between spatially adjacent voxels. In other words, each vertex in the octree is associated with one feature embedding. The feature embedding z_p is defined for any point $p \in \mathbb{R}^3$ in the space as $z_p = \psi(p; \mathcal{Z})$, where $\psi(.)$ is the tri-linear interpolation between corner features of the voxel in which p is contained. The collection of feature embeddings $\mathcal{Z}$ is learned to encode local properties of each voxel, while the tri-linear interpolation and shared features between adjacent voxels ensure a degree of continuity in the feature space.

Ray parameterization. Since we aim at learning voxel-based functions, we parameterize a given ray $\mathbf{r}$ in the local coordinates of each voxel. Specifically, as shown in Figure 5.3 (a), we compute the intersection points $p_1, p_2 \in \mathbb{R}^3$ of $\mathbf{r}$ with the minimum bounding sphere around the voxel, define vectors $\hat{p}_i$ from the voxel origin to these intersections, then scale them to unit norm: $\tilde{p}_i = \hat{p}_i / \|\hat{p}_i\|$, $i = 1, 2$. This gives us the overparameterized representation $\mathbf{r} = [\tilde{p}_1 \mid \tilde{p}_2] \in \mathbb{R}^6$.

5.4.2 Voxel-based light field function

The collection $\mathcal{Z}$ is split into feature embeddings $\mathcal{Z}^{\mathcal{T}}$ for optical thickness, and feature embeddings $\mathcal{Z}^{\mathcal{C}}$ for color. Each of these sub-collections has an associated small decoder network: $F^{\mathcal{T}}$ for optical thickness, and $F^{\mathcal{C}}$ for color.

Given a voxel v and a ray $\mathbf{r}$, we first compute the ray-voxel intersection points x_1, x_2 (see Figure 5.3 (b)) using an efficient ray-AABB intersection algorithm [228]. We then estimate the

optical thickness and color of the ray in two steps. The first step estimates whether the ray hits a surface within v, and produces an estimate for the location of the hit surface point $\hat{x}_s$ as a convex combination of x_1 and x_2. Specifically, we have:

$$\tau, \eta = F^{\mathcal{T}}(\mathbf{r}, z^{\mathcal{T}}), \tag{5.1}$$

where τ is the estimated optical thickness, $\eta \in [0, 1]$ is the mixing factor interpolating between x_1, x_2 , and $z^{\mathcal{T}} = [\psi(x_1; \mathcal{Z}^{\mathcal{T}}) \mid \psi(x_2; \mathcal{Z}^{\mathcal{T}})]$ is the combined interpolated feature embedding at the two intersection points. If $\tau \neq 0$, the surface point $\hat{x}_s$ is estimated as $\hat{x}_s = \eta x_1 + (1 - \eta)x_2$, where η can be seen as a within-voxel depth. Otherwise, if $\tau = 0$, the ray does not hit any surface. We estimate surface points to encourage continuous sampling of color features between adjacent voxels.

The second step samples color features $\mathcal{Z}^{\mathcal{C}}$ at the estimated surface point $\hat{x}_s$ and computes the color value c as

$$c = F^{\mathcal{C}}(\mathbf{r}, z^{\mathcal{C}}), \tag{5.2}$$

where $z^{\mathcal{C}} = \psi(\hat{x}_s; \mathcal{Z}^{\mathcal{C}})$ is the interpolated feature embedding at $\hat{x}_s$ (see Figure 5.3 (c)).

5.4.3 Rendering

To render a pixel from an SVLF, we perform volume rendering. We first evaluate the color c and optical thickness τ for all voxels intersected by the ray, then we aggregate the color values using alpha compositing.

Evaluating intersected voxels. We first compute a list of all voxels intersected by the ray. We use the sparse ray-octree intersection algorithm of [157] that leverages the hierarchical octree

structure and parallel scan kernels [229] for accelerated computation. This step yields x_1 and x_2 for each intersected voxel. We then evaluate the optical thickness τ and color c at each intersected voxel as described in § 5.4.2. Each voxel is evaluated only once by querying two lightweight decoder networks $F^{\mathcal{T}}$ and $F^{\mathcal{C}}$ to compute the optical thickness and color, respectively.

Aggregating color values along the ray. We use alpha compositing to compute the ray color $c(\mathbf{r})$ as:

$$c(\mathbf{r}) = \sum_{i=1}^{N} T_{v_i}(1 - e^{-\tau_{v_i}})c_{v_i}, \quad T_{v_i} = \prod_{j<i} e^{-\tau_{v_j}} \tag{5.3}$$

where N is the number of intersected voxels, T_{v_i} is the transmittance up until voxel v_i, and voxels are indexed by the order of their intersection along the ray.

5.4.4 Multi-stage training

We train SVLF in three stages:

1. Voxel training: We begin our training using surface rendering. For each ray, we use the corresponding depth to find the voxel containing the surface hit by the ray. We learn local features for each voxel to produce the correct color c, within-voxel depth η and optical thickness τ using the following loss:

$$\mathcal{L} = \sum_{v} \|c_v - c_{\text{gt}}\|^2 + \lambda_\eta \|\eta_v - \eta_{\text{gt}}\|^2 + \lambda_\tau \|e^{-\tau_v}\|^2 \tag{5.4}$$

   where the last term encourages predicting a zero transparency for the ray segment through v.

2. Volumetric training of optical thickness: Next, we freeze the color features $\mathcal{Z}^{\mathcal{C}}$ and decoder $F^{\mathcal{C}}$ (Eq. 5.2), and train the optical thickness network to predict the correct integral weights for volume rendering (Eq. 5.3). We use the same loss function of Eq. 5.4 except that we compute

the photometric loss with the aggregated ray color, $c(\mathbf{r})$, and we set λ_τ to zero to turn off direct supervision on the optical thickness τ. We observe that freezing the color network is important for convergence, otherwise the network gets stuck in poor local minima.

3. Fine-tuning: Finally, we unfreeze the color features $\mathcal{Z}^{\mathcal{C}}$ and decoder $F^{\mathcal{C}}$, and fine-tune the full model using the same loss function, but with a lowered learning rate.

5.4.5 Implementation details

Architecture. We represent a scene using a sparse octree, constructed at 128^3 voxel resolution. The octree implementation comes from the Kaolin library [230]. Specifically, we leverage the Structured Point Cloud (SPC) representation which supports efficient CUDA kernels for ray tracing. We have additionally implemented CUDA kernels for volumetric rendering using efficient scan primitives from CUB [229], which we leverage for fast and efficient rendering of our SVLF model. Both the optical thickness and color decoders $F^{\mathcal{T}}$, $F^{\mathcal{C}}$ are ReLU MLP networks. We use a single hidden layer for $F^{\mathcal{T}}$ and 3 hidden layers for $F^{\mathcal{C}}$, all with size 128. We apply a ReLU activation to the output optical thickness τ, and a sigmoid activation to both the mixing ratio η and color c outputs. We set the embedding size of the feature collections $\mathcal{Z}^{\mathcal{T}}$ and $\mathcal{Z}^{\mathcal{C}}$ to 64 and 32, respectively.

Training. We run our staged training (§ 5.4.4) for 300 epochs, which takes ∼160 minutes on a single NVIDIA V100 GPU. We use the Adam optimizer [231] with a learning rate of 1e-3 and a batch size of a single image down-scaled to a 400×400 resolution. We train using surface rendering for 100 epochs. We then freeze the weights of the color features $\mathcal{Z}^{\mathcal{C}}$ and decoder $F^{\mathcal{C}}$ and run the volumetric training stage for 150 epochs. Finally, we unfreeze the color weights

Table 5.3: Comparison of baseline methods on the 40-scene subset of our dataset at 400×400 resolution.

Method	Env.	Image-based metrics			Depth-based metrics	
		PSNR ↑	SSIM ↑	LPIPS ↓	RMSE (10^3) ↓	MAE (10^3) ↓
NeRF	Goog. Scan.	29.627±1.863	0.937±0.026	0.046±0.015	24.101± 14.234	43.854± 7.523
mip-NeRF	Goog. Scan.	31.727±1.681	0.952±0.019	0.032±0.010	18.726± 9.791	36.932± 7.464
Ins.-NGP	Goog. Scan.	**32.772**±1.985	**0.963**±0.012	**0.017**±0.007	**1.920**± 1.542	**7.515**± 2.784
SVLF	Goog. Scan.	30.868±1.941	0.954±0.020	0.019±0.009	9.200± 4.341	16.844± 12.567
NeRF	ABC	29.956±4.267	0.947±0.042	0.041±0.043	66.312±146.052	167.650±330.674
mip-NeRF	ABC	32.191±3.328	0.956±0.037	**0.024**±0.027	22.497± 11.307	41.285± 10.008
Ins.-NGP	ABC	**34.250**±4.564	**0.971**±0.027	0.025±0.024	**2.121**± 2.798	**14.230**± 5.016
SVLF	ABC	29.998±3.035	0.949±0.044	0.029±0.032	10.238± 7.008	18.816± 11.844
NeRF	Bricks	28.279±3.363	0.940±0.036	0.041±0.023	60.762± 43.947	34.122± 30.115
mip-NeRF	Bricks	31.329±3.361	0.956±0.030	0.023±0.014	40.941± 29.233	23.717± 6.869
Ins.-NGP	Bricks	**32.255**±3.277	**0.970**±0.019	0.023±0.013	**4.491**± 3.299	**7.977**± 3.023
SVLF	Bricks	29.244±3.265	0.949±0.032	**0.022**±0.014	20.300± 13.938	13.869± 9.300
NeRF	Amz. Ber.	25.778±3.704	0.796±0.092	0.184±0.108	28.490±192.051	74.041± 21.488
mip-NeRF	Amz. Ber.	**26.859**±3.231	0.784±0.088	**0.141**±0.077	14.919± 67.881	66.027± 17.342
Ins.-NGP	Amz. Ber.	25.405±5.657	**0.836**±0.089	0.161±0.103	**12.693**±112.870	**36.202**± 33.896
SVLF	Amz. Ber.	25.197±3.936	0.795±0.103	0.161±0.106	15.305± 49.771	38.119± 48.792
NeRF (0.15 fps)	all	28.363±3.819	0.905±0.049	0.078±0.046	44.916± 99.071	79.917± 97.450
mip-NeRF (0.3 fps)	all	30.526±2.900	0.912±0.043	**0.055**±0.032	24.271± 29.090	41.990± 10.421
Ins.-NGP (60 fps)	all	**31.170**±3.871	**0.935**±0.037	0.056±0.037	**5.306**± 30.127	**16.481**± 11.180
SVLF (12.10 fps)	all	28.827±3.044	0.912±0.050	0.058±0.040	13.761± 18.764	21.912± 20.626

($\mathcal{Z}^{\mathcal{C}}$, $F^{\mathcal{C}}$), and continue volumetric training for another 50 epochs with a learning rate of 2e-4.

5.5 Experiments

In this section, we benchmark different baselines against our dataset on two main tasks: single scene view synthesis and few-shot view synthesis. We evaluate the view reconstruction quality of baselines using PSNR and SSIM (higher is better) and LPIPS [232] (lower is better). We also evaluate the reconstructed depth maps using RMSE and MAE (lower is better).

5.5.1 Single scene view synthesis

In our dataset, each scene is composed of 150 unique views. We propose to use 100 views for training, 5 views for validation, and 45 views for testing. For this task we evaluate four base-

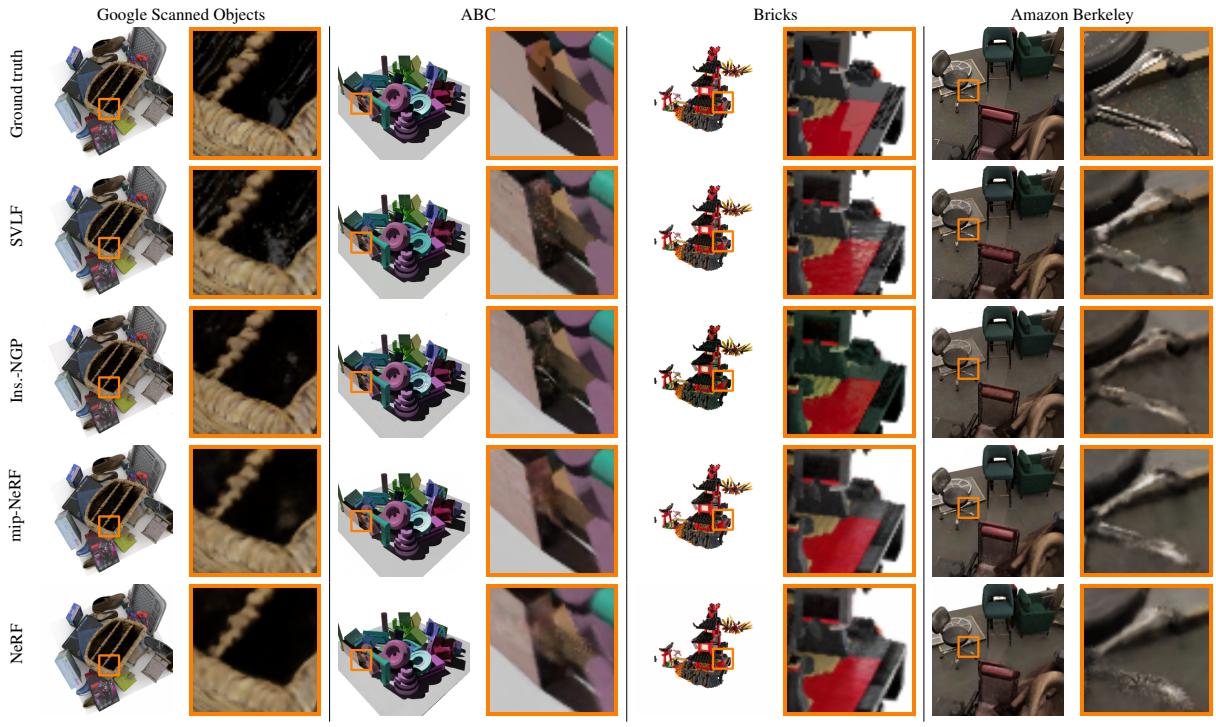


Figure 5.4: Single-scene novel view synthesis results of the baselines on a sample scene from each environment.

lines: NeRF [40], mip-NeRF [149], the concurrent work of Instant-NGP [4] and our proposed SVLF. We use the author-provided implementations for NeRF[2], mip-NeRF[3], and Instant-NGP[4] and train all methods until convergence. The number of training steps was determined by the saturation of the mean PSNR value over the validation set for a few scenes and then was fixed for all scenes. As NeRF and mip-NeRF have quite large training time, we propose a smaller subset composed of 10 scenes per environment, for a total of 40 scenes. Finally, we mask out the background in the Amazon Berkeley environment as the baselines are not designed to handle complex backgrounds.

Table 5.3 presents high-level results for image and depth-based metrics, and Figure 5.4 presents qualitative comparisons. All methods perform reasonably well on the Google Scanned

[2] https://github.com/bmild/nerf
[3] https://github.com/google/mipnerf
[4] https://github.com/NVlabs/instant-ngp

Table 5.4: Results for the full dataset at 1600×1600 resolution with Instant-NGP [4].

Env.	Image-based metrics			Depth-based metrics	
	PSNR ↑	SSIM ↑	LPIPS ↓	RMSE (10^3) ↓	MAE (10^3) ↓
Goog. Scan.	31.269 ± 1.461	1.486 ± 0.014	0.054 ± 0.013	0.634 ± 0.591	6.790 ± 2.829
ABC	33.888 ± 3.842	0.975 ± 0.017	0.034 ± 0.024	1.311 ± 2.075	13.913 ± 5.104
Bricks	31.170 ± 3.281	0.966 ± 0.020	0.045 ± 0.029	2.526 ± 8.298	7.863 ± 8.112
Amz. Ber.	23.129 ± 4.882	0.851 ± 0.078	0.254 ± 0.097	17.766 ± 164.130	47.861 ± 53.498
all	30.358 ± 3.334	1.030 ± 0.027	0.077 ± 0.036	4.412 ± 30.355	14.854 ± 13.878

Objects and the Bricks environments which have a setup similar to the NeRF Synthetic dataset [40]. NeRF and SVLF performance degrade on the ABC environment, and none of the baselines performs well on our most challenging environment, Amazon Berkeley, with Instant-NGP being the most robust of the baselines, followed by mip-NeRF as it is trained on multi-scale images. SVLF benefits from depth supervision during training and therefore constructs better depth maps than NeRF and mip-NeRF. In our experiments we have observed that SVLF does not always capture view dependent effects due to the lack of global context in our voxel-based formulation and the absence of an explicit view direction input.

Finally, we show results using Instant-NGP for all 1927 scenes of the dataset at full resolution in Table 5.4.

Runtime comparison. We compare the training and rendering speed of SVLF to the baselines in Table 5.5. Training is done on a single V100 GPU, while inference is done on a single RTX 3090 GPU. SVLF trains an order of magnitude faster than the other baselines as it utilizes depth maps during training. This observation is also consistent with [152], a depth-supervised NeRF.

To compare inference time, we compute the average rendering time over the 45 test images of the first scene of the Google Scanned Objects environment. SVLF is over 80x faster to render than NeRF, and approximately 40x faster than mip-NeRF on our test scene. There are two main

Table 5.5: Training and inference time comparison on the single-scene novel view synthesis task.

Method	Training		Rendering	
	# steps	time (hrs)	time/frame (sec)	FPS
NeRF	500K	26.3	6.86	0.15
mip-NeRF	500K	35.0	3.30	0.30
SVLF	30K	2.7	0.08	12.10

reasons for this speedup. First, rendering SVLF requires significantly fewer network queries per ray compared to NeRF. The number of queries is determined by the number of intersected voxels along the ray, which is efficiently computed using the ray-octree intersection algorithm of [157]. On average, we have less than 10 voxel evaluations per ray on our scenes. Second, the cost per query is much cheaper for SVLF. This is due to moving much of the network capacity to the learned local features, and thus enabling using a much smaller decoder architecture ($\sim$16x smaller) than the NeRF network.

5.5.2 Few-shot view synthesis

Given that each environment includes multiple scenes with shared characteristics, camera distributions, object types, and/or lighting, we can use any environment to train few-shot view synthesis algorithms. Different methods [9, 166] have shown that a single or few views (< 5) can be used to predict new views. We propose to partition our dataset as follows: 280 scenes for training, 2 scenes for validation, and 18 scenes for testing. Since the Bricks environment is bigger, we propose to use 900 scenes for training, 30 for validation, and 100 for testing. We use the freely available training code[5] for PixelNeRF [9]. We found the algorithm unstable to train, especially when having random camera poses, and eventually diverging after a few hundred

[5] https://github.com/sxyu/pixel-nerf

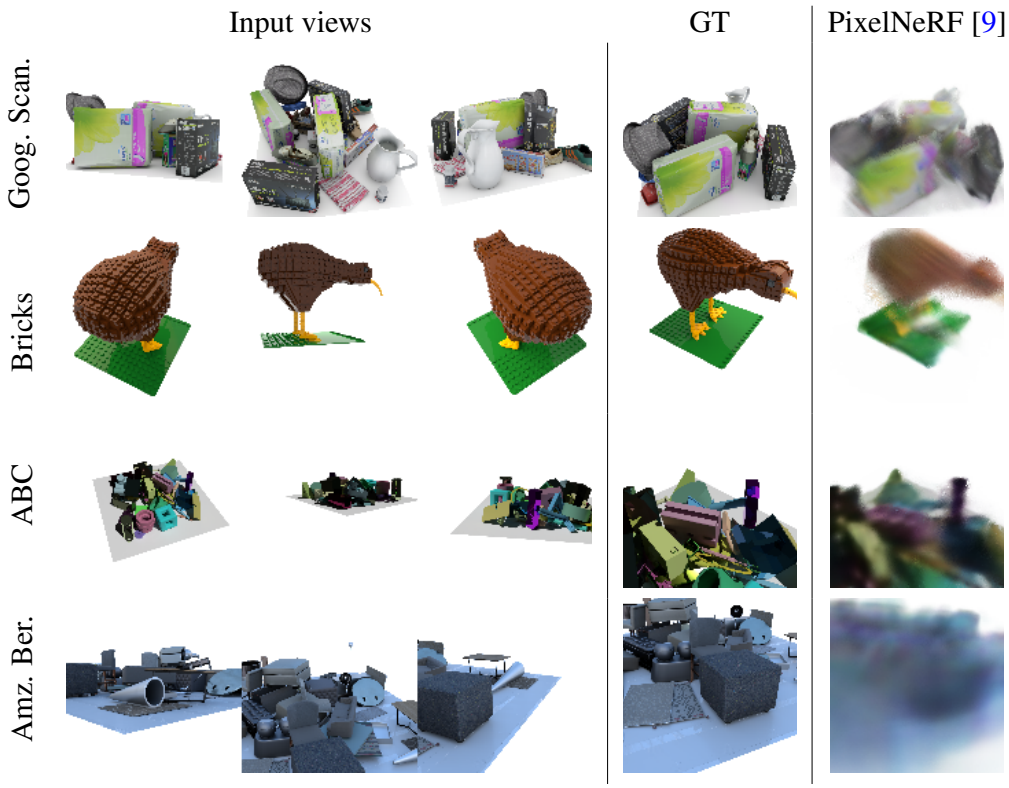


Figure 5.5: Sample results of PixelNeRF [9] on our dataset.

epochs. We terminate the training when the model diverges and use the latest proper checkpoint for evaluation. We show qualitative results for a 3-shot input experiment in Figure 5.5 and report quantitative metrics in Table 5.6. Even though the results are encouraging, it is clear that future work is needed in the area of few-shot view synthesis.

Table 5.6: Pixel-NeRF results.

env.	PSNR ↑	SSIM ↑
Goog. Scan.	14.588	0.483
ABC	12.149	0.629
Bricks	12.149	0.523
Amz. Ber.	12.126	0.318
all	12.753	0.488

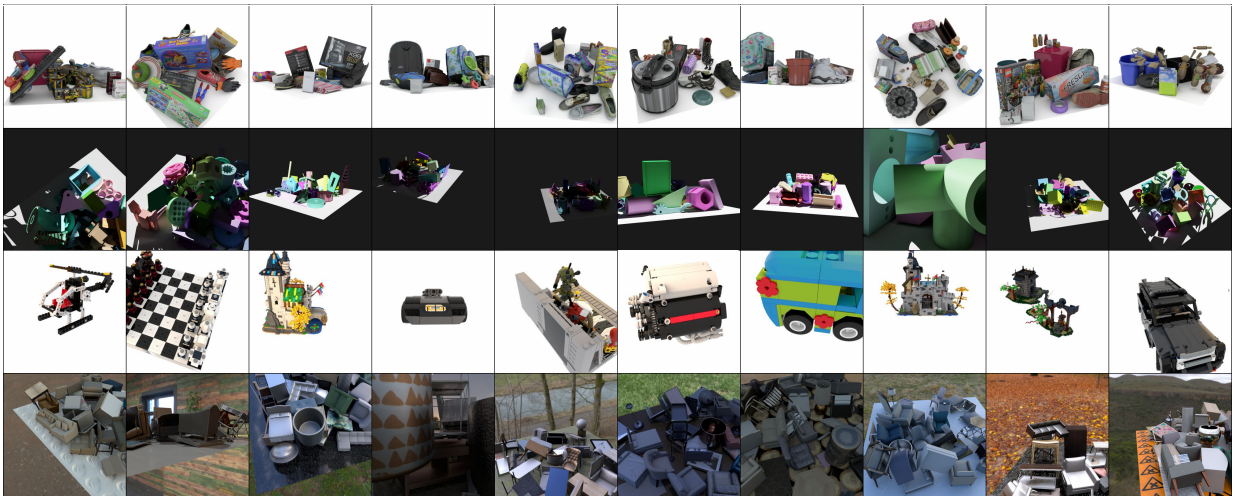


Figure 5.6: Examples of multiple views from different environments: Google Scanned (1^{st} row), ABC (2^{nd} row), Bricks (3^{rd} row), and Amazon-Berkeley (4^{th} row). (Best viewed when zoomed.)

5.6 Additional results and discussion

5.6.1 Data generation

Camera pose distribution. Our RTMV dataset is designed to be more challenging than existing novel view synthesis datasets. In addition to its diversity in terms of scene lighting and objects materials and textures (see Figure 5.6), it contains two types of camera placements: hemisphere and free. The 'hemisphere' camera placement is an easy setup where cameras are placed at an equal distance from the scene (*i.e.*, on a hemisphere). Therefore, trained multi view synthesis models do not need to reason about multi-scale. In the 'free' camera placement, the camera is allowed to roam freely within a cube surrounding the scene, thus providing a variety of multi-scale images of the scene. Figure 5.7 shows the distribution of camera azimuth and elevation for each of our four environments.

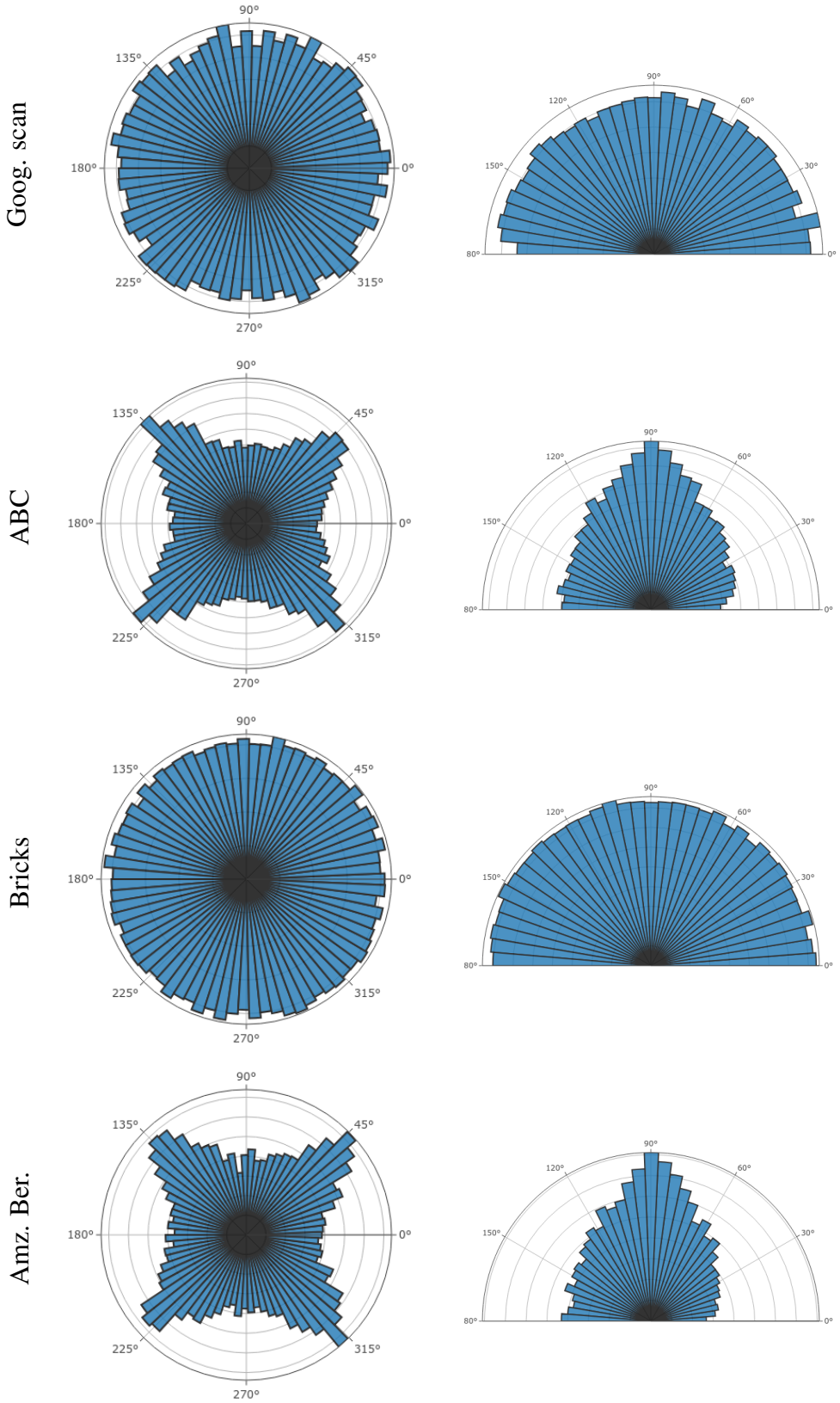


Figure 5.7: Camera azimuth and elevation distributions for our different environments

```python
import raytracer
raytracer.initialize()
# Create camera
my_camera = raytracer.entity.create(
    name      = 'cam',
    transform = raytracer.transform.create('c_tfm'),
    camera    = raytracer.camera.create('c_cam')
)
my_camera.get_transform().look_at(
    eye = [3, 3, 3], at = [0, 0, 0], up = [0, 0, 1]
)
raytracer.set_camera_entity(my_camera)
# Create object
my_object = raytracer.entity.create(
    name      = 'obj',
    transform = raytracer.transform.create('o_tfm'),
    mesh      = raytracer.mesh.create_sphere('o_mesh'),
    material  = raytracer.material.create('o_mat')
)
raytracer.material.get('o_mat').set_base_color([1, 0, 0])
# Render image
raytracer.render_to_file(
    width = 512, height = 512,
    samples_per_pixel = 1024,
    file_path = 'image.png'
)
raytracer.deinitialize()
```

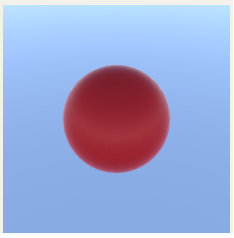


Figure 5.8: A minimal Python script that renders the inset image.

Python-based ray tracer. Our Python-based ray-tracing renderer is built on NVIDIA OptiX [217] with a C++/CUDA backend. It uses path tracing to render high-quality and physically-based images. In addition, it appeals to non-graphics experts through its simple Python API. The scripting capability makes it easy to install, use and share. Figure 5.8 shows a simple code example highlighting the simplicity of our Python API.

5.6.2 Additional qualitative results

We show more qualitative results on each of our four environments in Figures 5.11, 5.13, 5.15 and 5.17, and the corresponding depth maps in Figures 5.12, 5.14, 5.16 and 5.18. SVLF trains with depth supervision, and thus constructs better depth maps compared to NeRF and mip-NeRF. Instant-NGP gives the best results but suffers from floating artifacts which can also be seen in the recovered depth maps. All baselines perform reasonably well on the Google Scanned Objects and the Bricks environments. However, the performance of both NeRF [40] and SVLF drops on the ABC environment due its challenging camera poses and lighting conditions. And

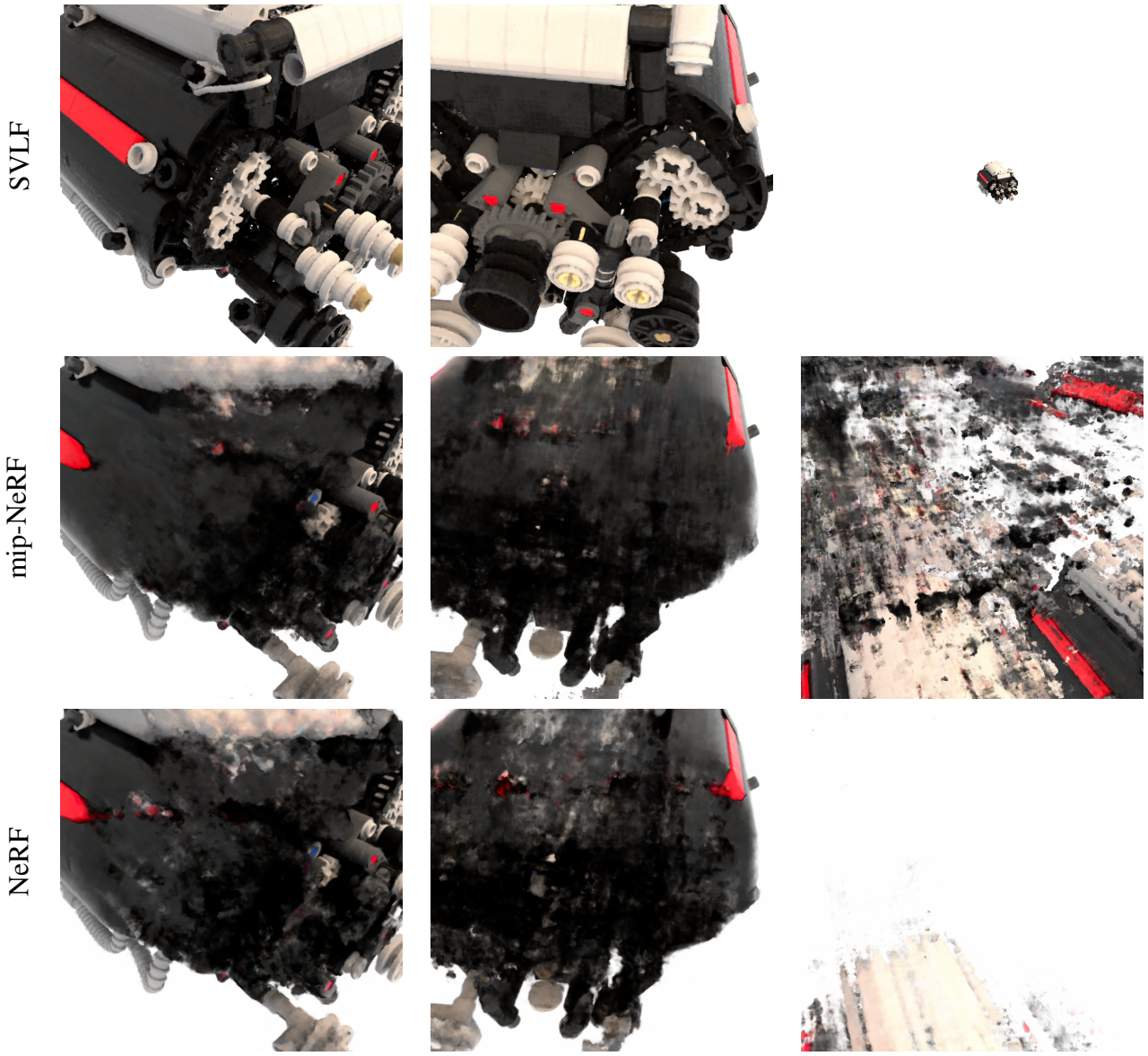


Figure 5.9: The render planes parameterization of the baselines can lead to sub-optimal results when the camera moves freely in the space.

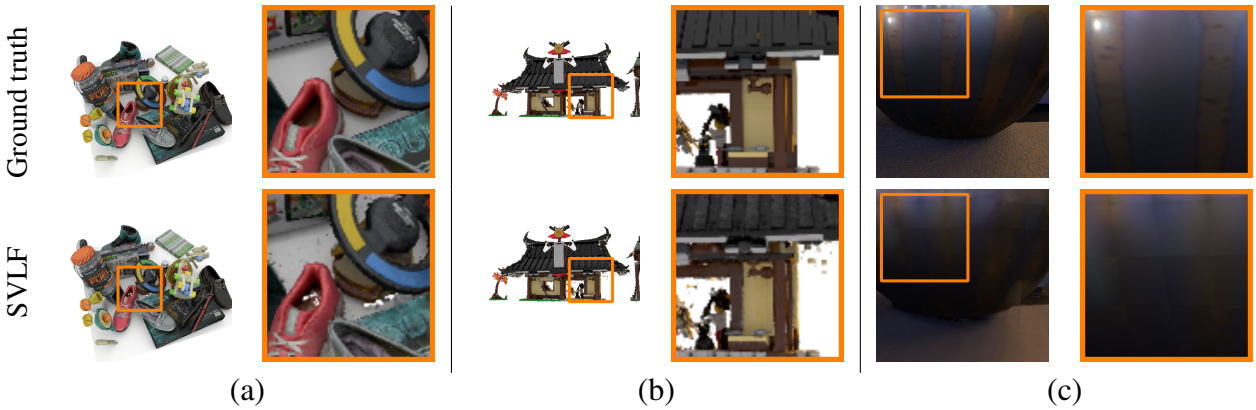


Figure 5.10: Example limitations of SVLF. (a) and (b) show occasional floater/hole artifacts due to sub-optimal training of the optical thickness τ. (c) shows seams between voxel boundaries when the camera is too close to the scene, and a lack of specular highlights (figure best seen in zoom).

eventually, all baselines struggle on our most challenging environment (Amazon Berkeley). The multi-scale training of mip-NeRF [149] makes it more robust to random camera poses compared to the other baselines, especially on the Amazon Berkeley environment. Instant-NGP suffers from floating artifacts especially with random camera poses (*e.g.*, ABC and Amazon Berkeley environments). But overall, Instant-NGP recovers fine details, renders sharp results, and achieves the best performance on our dataset.

The voxel-based formulation of SVLF allows for a better representation of the scene that accommodates free-moving cameras. This is because SVLF focuses on rays intersecting voxels, rather than where rays originate. In contrast, the near and far render planes parametrization of NeRF and mip-NeRF poses a limitation. On one hand, setting near and far planes to tightly bound the scene leads to better point sampling along the ray during training, but allows for less flexibility for moving the camera at render time. On the other hand, setting the near and far plane to loosely enclose the scene allows for more flexible camera movement at inference time, but leads to poor training quality as NeRF and mip-NeRF are sensitive to how densely points are

sampled along the ray. Figure 5.9 highlights this limitation by showing example results for when the camera is placed at very close (first two rows) or far (third row) distance from the scene.

5.6.3 SVLF limitations

In this section we discuss some of the limitations of the proposed SVLF approach.

Voxel resolution. SVLF uses a sparse octree to represent a voxel grid of size 128^3. While this voxel resolution allows for high quality results over the test set of our scenes, it can produce voxel boundary artifacts if the camera zooms in too close to the scene. Specifically, the seams between voxels become more visible as the camera zooms in (see Figure 5.10).

Sub-optimal training of optical thickness. SVLF learns voxel-based functions. Although using a shared decoder as well as the tri-linear interpolation between voxel features provide a degree of continuity between adjacent voxels, voxel features could still suffer from a degree of independence. This especially happens to the optical thickness features, $\mathcal{Z}^{\mathcal{T}}$, where the tri-linear interpolations happens at the intersection points, (x_1, x_2), between the ray and the voxel, rather than at the estimated surface point $\hat{x}_s$. This could lead to a lack of global context between voxels along the ray, thus resulting in errors in the learned optical thickness τ. Figure 5.10 show example failures where incorrect predictions of the optical thickness τ lead to either floaters or holes in the rendered image. Note the holes inside and beside the red shoe in Figure 5.10-(a), and random floaters in Figure 5.10-(a), (b).

Translucent surfaces. SVLF evaluates the ray color within a voxel at a single point location (the estimated surface hit $\hat{x}_s$). Therefore, we assume solid surfaces, and bootstrap the training using surface rendering (§ 5.4.4). While SVLF utilizes a volumetric training stage to learn non-binary

optical thicknesses/densities, the support for translucent surfaces remains limited due to sampling a single point within each voxel.

View-dependent effects. We observe that SVLF does not capture view dependent effects very well (*e.g.*, specular highlight on the vase in Figure 5.10-column 3.) We hypothesize this is due to the absence of an explicit view direction input to our network. Adding the view direction as an extra input besides the estimated surface hit $\hat{x}_s$ could help mitigate the problem.

5.7 Conclusion

In this chapter, we proposed RTMV, a large-scale, high quality and ray-traced synthetic dataset for novel view synthesis. Our dataset is orders of magnitude larger than currently used datasets and offers a challenging variety in terms of camera poses, lighting conditions, object material and textures. Thus, RTMV is suitable as a demanding benchmark to evaluate view synthesis algorithms, which can help advance novel view synthesis research. We also proposed SVLF, a voxel-based accelerated algorithm for novel view synthesis. SVLF is an order of magnitude faster to train on synthetic images, and two orders of magnitude faster to render than NeRF, while retaining similar output quality. SVLF achieves such a speedup by leveraging depth maps to build an explicit/implicit hybrid representation that combines a coarse and sparse voxel octree with learned local features to render high quality results.

Figure 5.11: More qualitative results on the Google Scanned Objects environment (figure best seen in zoom).

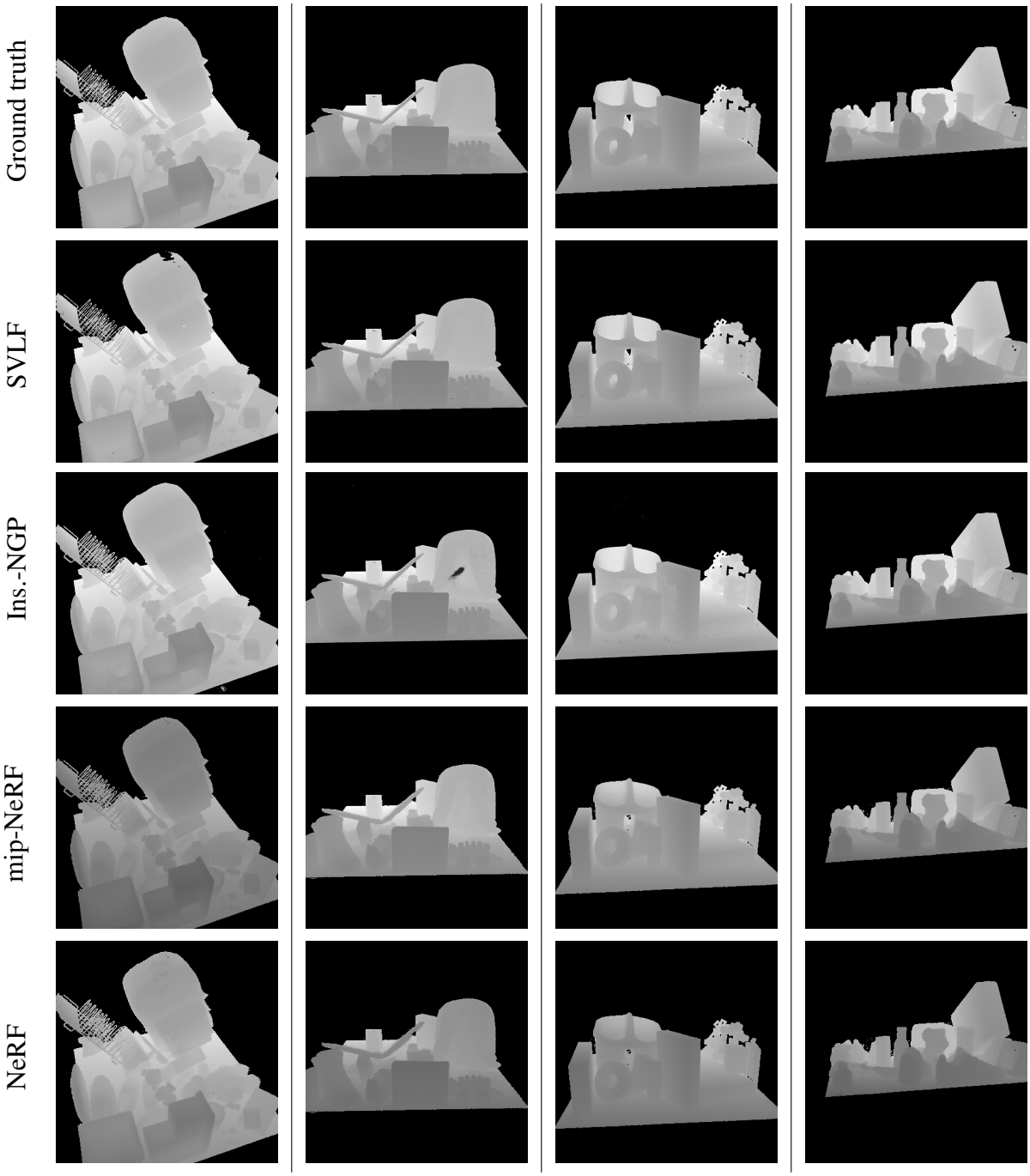


Figure 5.12: Depth qualitative results on the Google Scanned Objects environment (figure best seen in zoom).

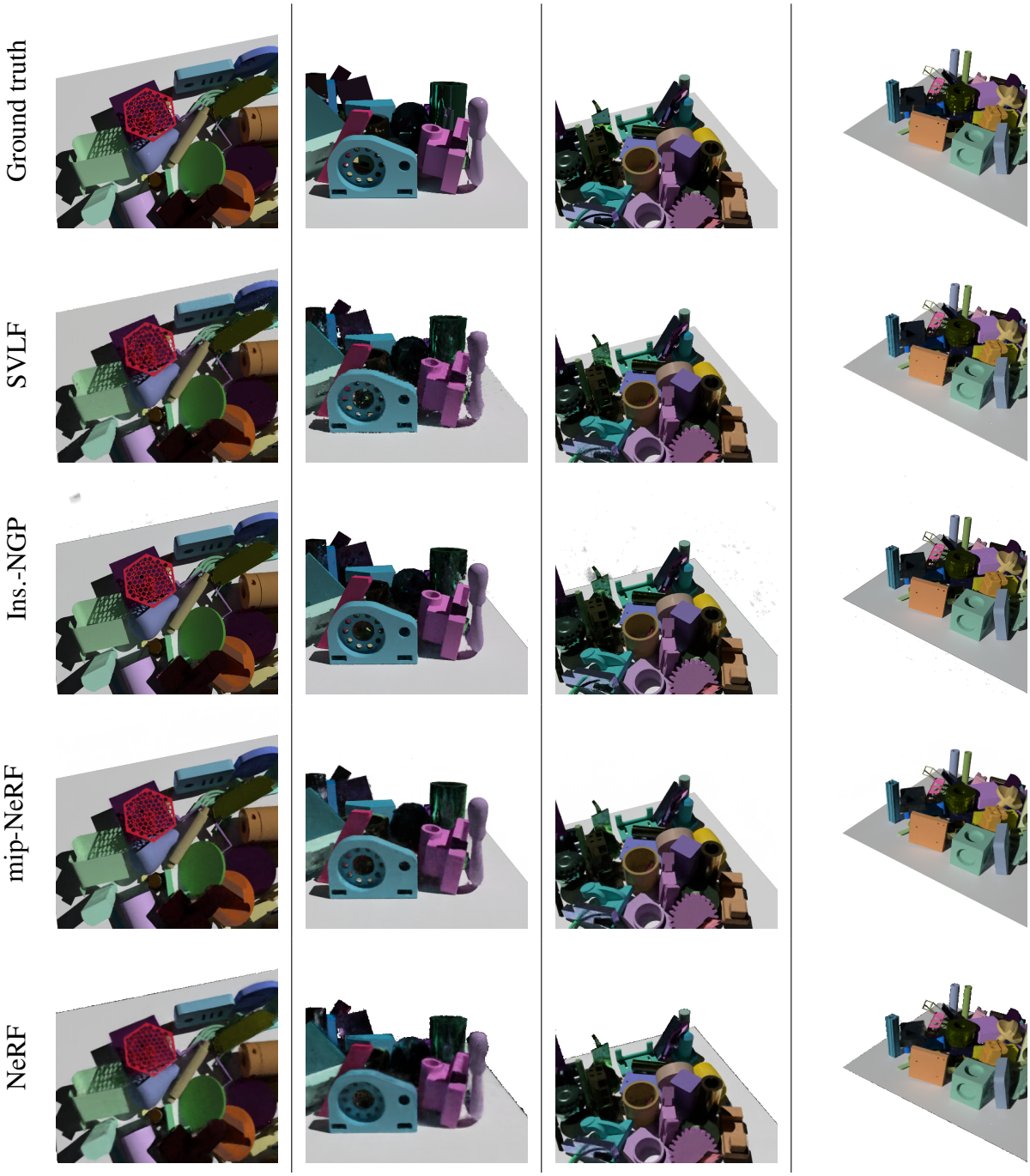


Figure 5.13: More qualitative results on the ABC environment (figure best seen in zoom).

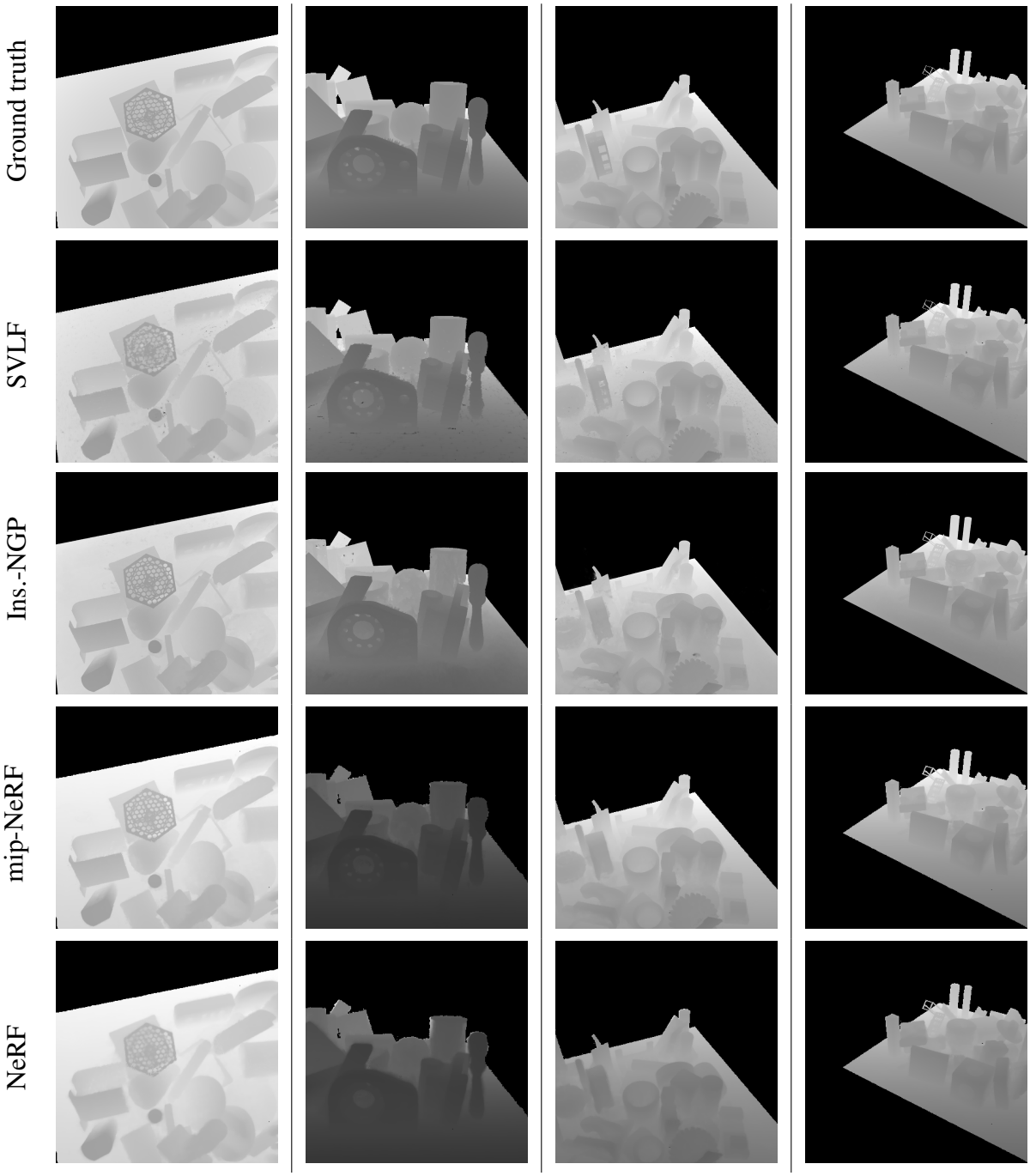


Figure 5.14: Depth qualitative results on the ABC environment (figure best seen in zoom).

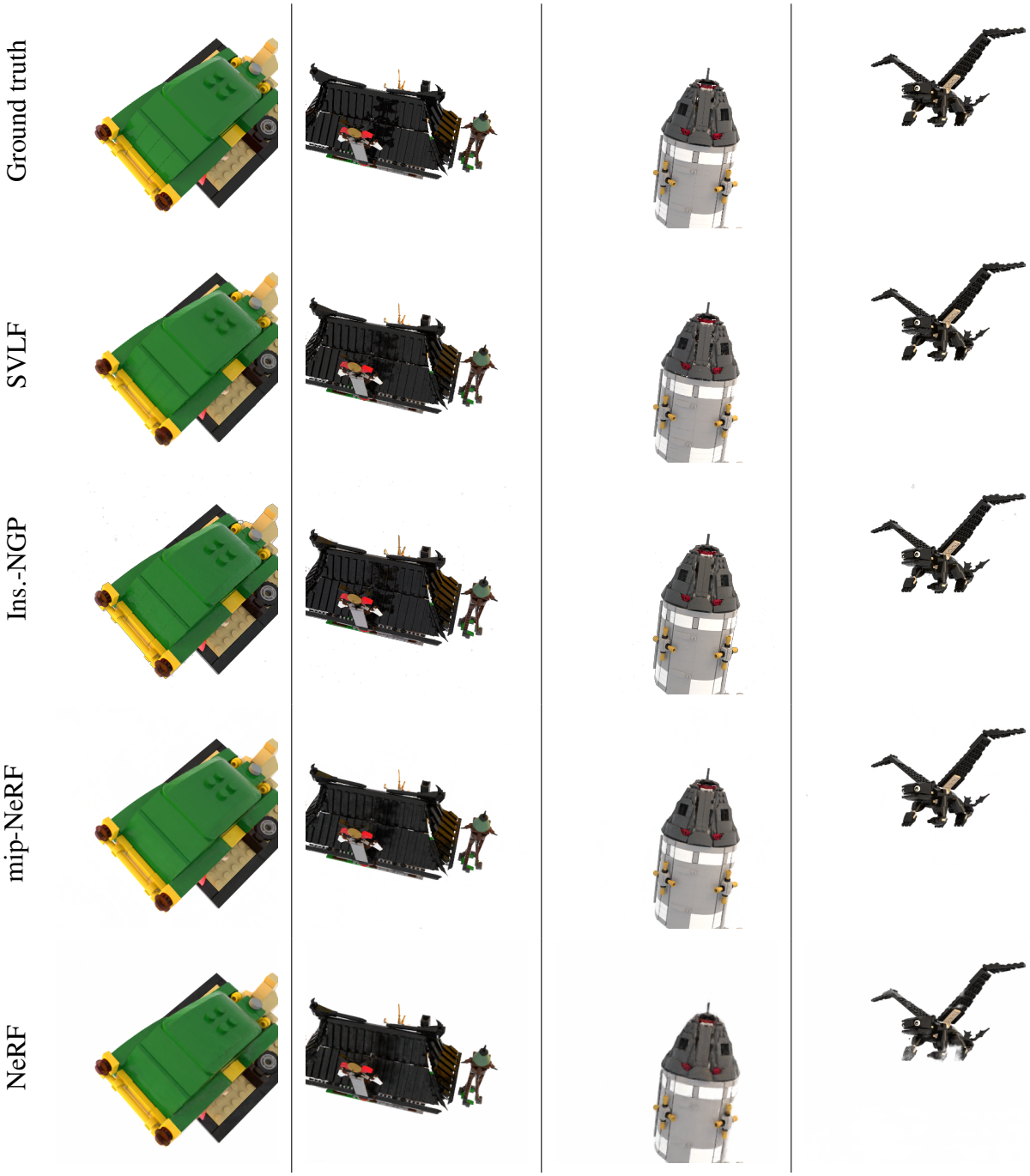


Figure 5.15: More qualitative results on the Bricks environment (figure best seen in zoom).

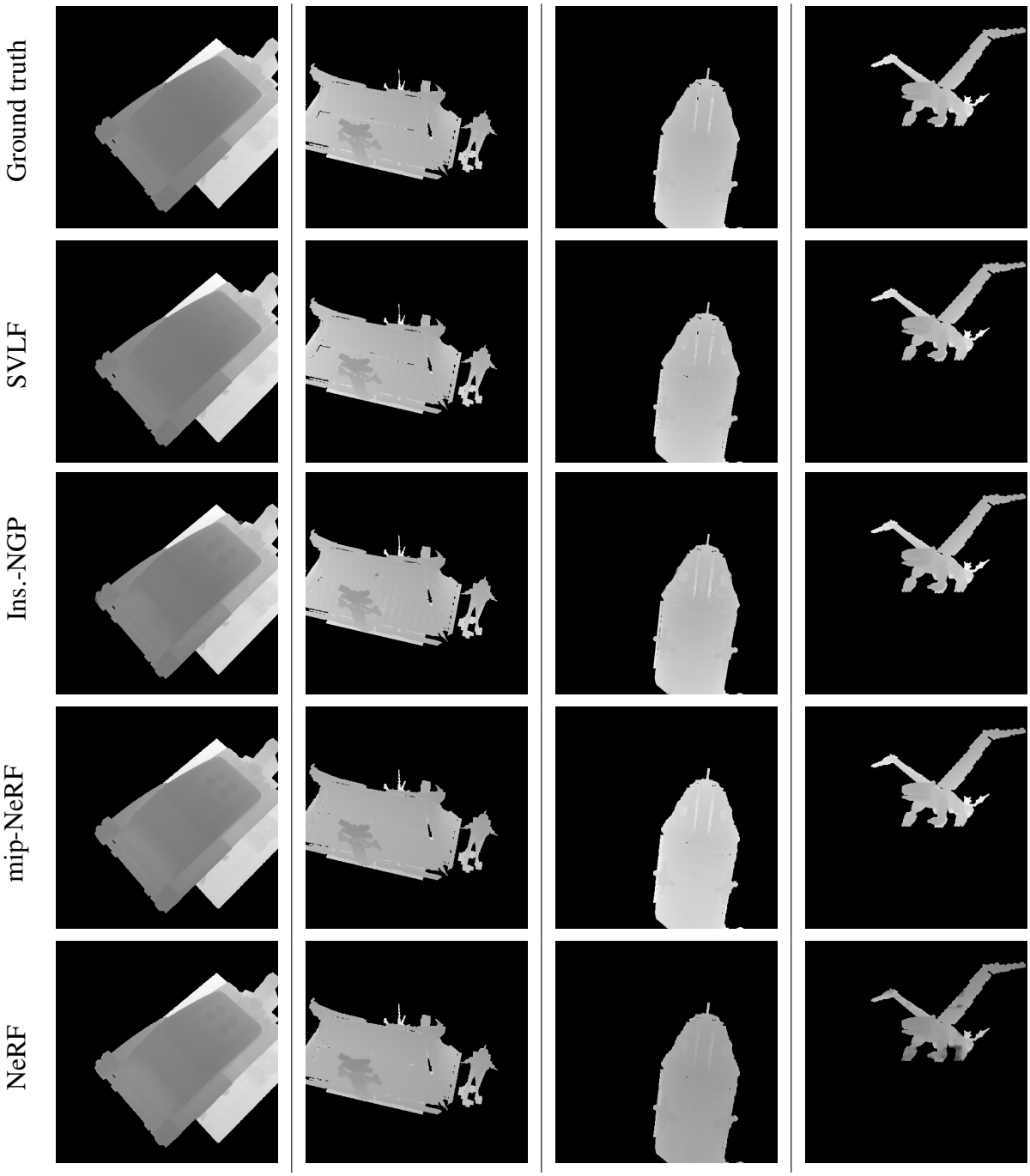


Figure 5.16: Depth qualitative results on the Bricks environment (figure best seen in zoom).

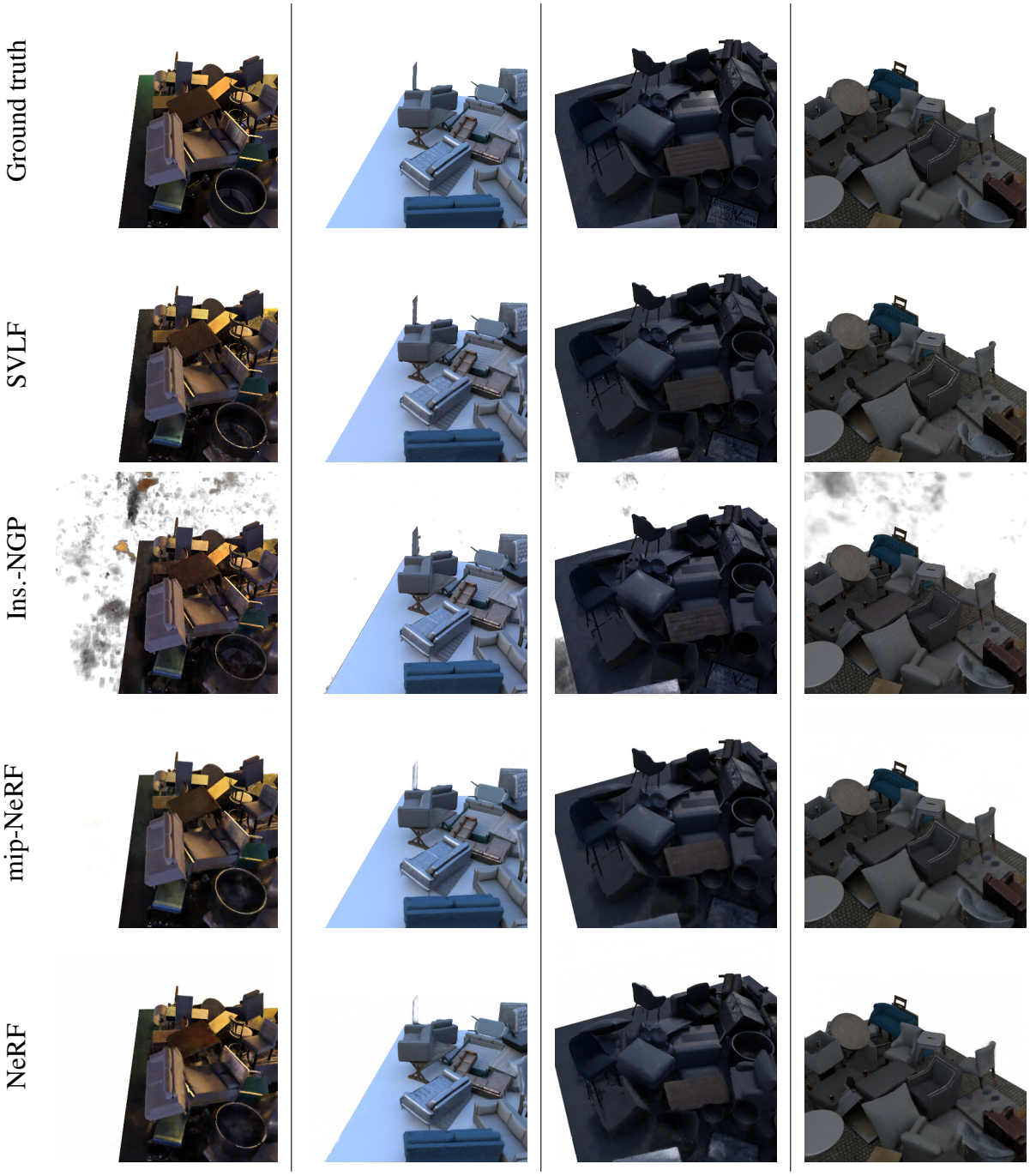


Figure 5.17: More qualitative results on the Amazon Berkeley environment (figure best seen in zoom).

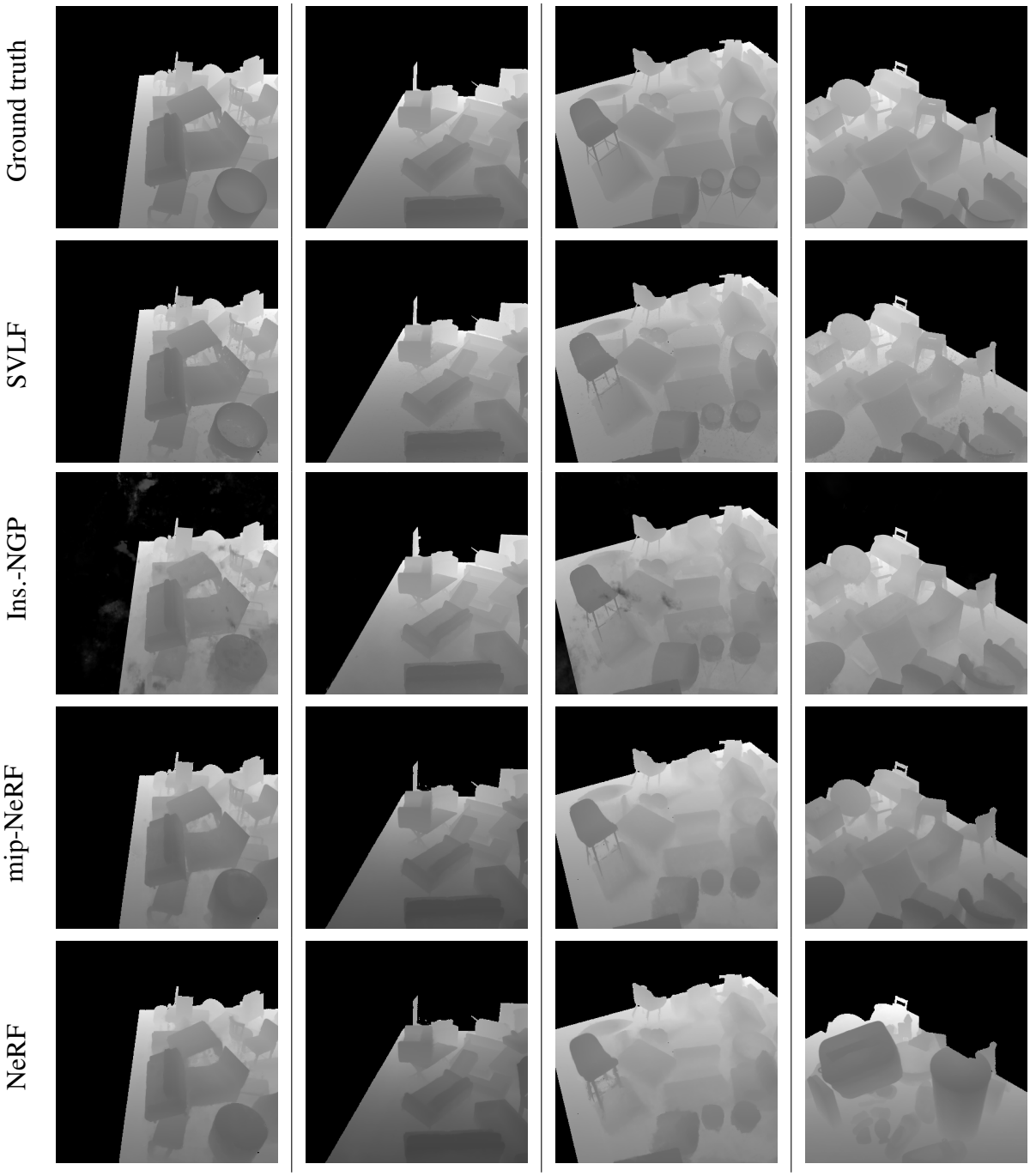


Figure 5.18: Depth qualitative results on the Amazon Berkeley environment (figure best seen in zoom).

Chapter 6: Conclusion and Open Challenges

To conclude, we discuss the contributions of this dissertation to the task of photo-realistic image synthesis and its applications to the problem of novel view synthesis. Then we finish by discussing some of the open challenges and future research directions. Then we discuss some of the ethical concerns for photo-realistic image synthesis, and we finish by discussing some of the open challenges and future research directions.

6.1 Summary

This dissertation started by addressing the multi-modal nature of image-to-image (I2I) translation frameworks for photo-realistic image synthesis. In Chapter 2, we proposed *StEP*, a style-based encoder pre-training strategy to model the distribution of output styles (in terms of color and texture) in I2I translation. In contrast to training a domain-specific style latent space jointly with the I2I translation network, the proposed style pre-training can be done only once using auxiliary data, and it generalizes well to many image domains. This allows for faster and more stable training of I2I translation networks with fewer loss terms, and yet achieves more faithful style capture and transfer compared to prior state-of-the-art.

While I2I translation introduces a degree of user control over the image synthesis process of GANs, such control is mainly in 2D, as I2I translation maps between images (2D) from an input

and output domains. In this dissertation, we extended the user control to 3D by conditioning the rendering process on an a coarse-explicit or learned-implicit 3D representation of target objects or scenes. In specific, we applied Neural Rendering techniques to solve the problem of novel view synthesis (NVS), under different challenging settings. In Chapter 3, we utilized point clouds as a coarse geometric proxy and scaled up NVS to in-the-wild internet photos of tourist landmarks. Our data-driven approach successfully filters out transient occluders (*e.g.*, tourists and other objects) from internet images, and renders photo-realistic images of a target tourist landmark from novel views and under different appearance (*e.g.*, lighting, time-of-day, and weather conditions). In Chapter 4, we proposed LSR, a framework that learns a latent spatial representation for novel view synthesis of human heads in the challenging single-shot/few-shot setting. We used sparse facial landmarks and a learned dense spatial representation as a proxy to render any target subject under different head poses and facial expressions. LSR generalizes to new test subjects that were not part of the training, and generates photo-realistic and identity-preserving results given as few as a single image of the target subject. Finally, in Chapter 5, we used a coarse voxel grid as a geometric proxy and augmented it with learned local features to enable high-quality novel view synthesis of synthetic scenes. Specifically, we learned a voxel-based light field framework that is an order of magnitude faster to train on synthetic scenes compared to traditional Neural Radiance Fields [40] (NeRF), and two orders of magnitude faster to render while maintaining similar output quality to NeRF. In addition, we constructed and released RTVM, a large scale, high quality and ray-traced synthetic dataset for novel view synthesis. The RTMV dataset is orders of magnitude larger than currently used synthetic datasets for NVS, and offers a challenging variety in terms of camera poses, lighting conditions, and object materials and textures. Thus, it is suitable as a challenging benchmark to evaluate view synthesis algorithms and unravel their limitations,

which can help advance novel view synthesis research.

6.2 Ethical concerns and media forensics

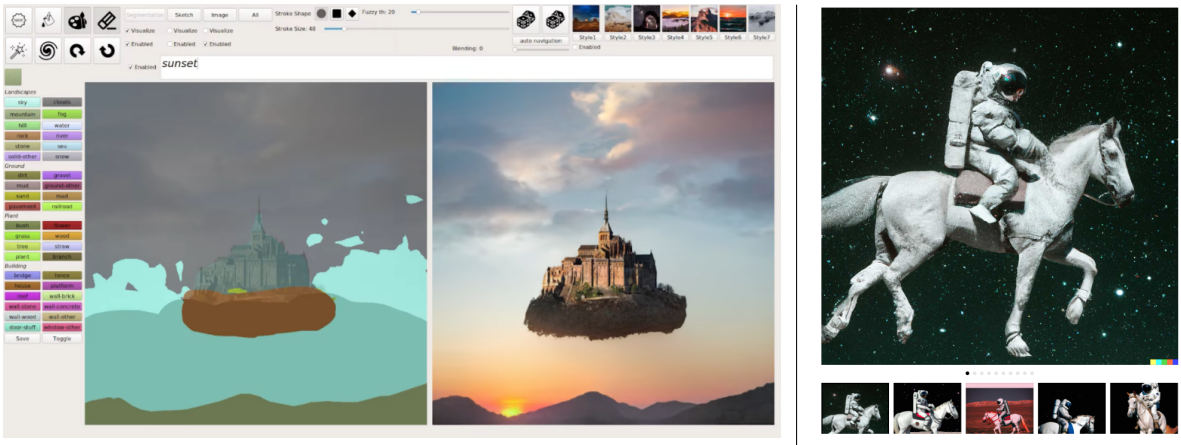
Photo-realistic image and video synthesis opens up the door to many exciting opportunities and applications. For example, telepresence, augmented reality, low-bandwidth high-quality communication, motion capture and animation and visual effects are some of applications that already benefit greatly from advances in photo-realistic image and video synthesis. However, there is a growing concern about the negative impact of potential malicious uses of such technologies. Most notably, facial reenactment and deepfake techniques can be used to generate harmful fake media. Therefore, it is important to develop fake detection systems and implement such systems in various media distribution platforms to detect and filter out harmful fake content. While current deepfake detection systems can detect fake images and videos with high classification accuracy and with high confidence, they rely in no small part on detecting artifacts and fingerprints specific to the respective architectures of the generators (*e.g.*, artifacts of CNN-based decoders, or artifacts around face composition for face-swap and other deepfake techniques). Therefore, such fake detection systems can overfit the artifacts of certain architectures and fail to generalize to new fake generators. For example, other successful architectures that are significantly different from CNN-based decoders include Diffusion Models [233, 234, 235], Auto-regressive Models [62, 236, 237, 238, 239], and Coordinate-based Networks [37, 38, 39, 40, 41]. And with quick pace with which photo-realistic image and video synthesis research is advancing, and with the increasingly fast expansion in the number of fake generators, there is an urgent need to improve and update fake detection systems at an equal, if not faster, pace.

Figure 6.1: Recent high quality 3D synthetic datasets bridge the gap with real datasets. (Figure adopted from [10]).

During the course of different works of this dissertation, we contributed to the DARPA Media Forensics (MediFor) and DARPA Semantic Forensics (SemaFor) programs, by generating and facilitating the generation of training data for fake detection systems, as well as train/fine-tune existing fake detection systems to counter potential misuses of such technologies. It is important to continue research in the field of fake image/video detection to keep on par with the ever improving field of image/video synthesis, as not only do models improve, but also the ease of access to such models grows rapidly.

6.3 Open challenges and future directions

One main limitation to using Generative Adversarial Networks (GANs) for image synthesis is that they mainly operate in 2D. While this dissertation proposed to combine GANs with rough 3D proxies to get around this limitation, the quality of the output is still limited by the quality of the recovered 3D proxy. For example, in Chapter 3, we showed that our model generates



(a) NVIDIA's GauGAN2 [245]

(b) OpenAI's DALL-E 2 [246]

Figure 6.2: Examples for using AI to create art by combining generative models with the imaginative power of human artists.

blurry output when the input point cloud rendering is very sparse. On the other hand, implicit neural representations (*e.g.*, [37, 38, 39, 40, 41]) can model arbitrary geometry, in addition to their high quality synthesis and multi-view consistency. Therefore combining GANs with implicit neural representations emerged recently as an exciting new avenue for 3D-aware generative modeling [193, 240, 241], and has been advancing at an incredible pace [194, 242, 243, 244].

Some of the clear extensions to rising 3D GANs can be listed by drawing parallels between them and their 2D counterparts. For example, to introduce more user control over the the synthesis process, we need a conditional version of 3D GANs, where the output is conditioned on some user-defined input. This conditional input can be an image or a few images of the target object, an artist sketch of the object, or even a text description of the object. Another direction is to gain semantic control over the synthesis process, such as controlling the shape and/or style of different semantic regions like hair, skin, and wearable accessories for face synthesis. But in general, we need 3D GANS to learn a disentangled representation that would enable controlling lighting, surface material and other factors of variation of the scene.

Another open challenge is the lack of multi-view consistency in 3D GANs, which are especially clear when rotating the camera pose or synthesizing a video. These multi-view consistency errors arise from using a neural network, which is essentially a black box decoder, to render the final image. Rendering images directly using a neural network does not offer control over fine features (*e.g.*, fine texture details), which often impairs multi-view consistency and causes flickering artifacts between different views. This multi-view inconsistency thus limits the use of 3D GANs for applications with low error tolerance, such as AR/VR and simulations for self-driving cars. One possible direction to tackle this problem is to have 3D GANs generate intermediate scene representations, such as a textured mesh, which is then rendered using well-understood graphics pipelines. As a starting step, we can use high quality synthetic 3D data (*e.g.*, Figure 6.1) to explicitly supervise learning these disentangled intermediate representations. But an even more exciting direction would be to use promising works on differentiable mesh rendering [247] to learn this disentangled representation purely from data.

In the end, we have seen advances in data-driven generative models, such as GANs and more recently Diffusion Models [233, 234, 235], open up the door for many creative applications by combining their generative power with the creative power of the human imagination. Figure 6.2 shows recent examples for the potential of using AI to create art. For example, NVIDIA'as GauGAN [133, 245] enables creative content creation by painting with simple semantic constructs and borrowing styles from real images. Another example is OpenAI's DALL-E [246, 248], which enables creative content creation from text descriptions. While these exhilarating creative works generate art in 2D, they inspire achieving a similar vision in 3D by building on top of recent advances in photo-realistic and 3D-aware generative models.

Bibliography

[1] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 586–595, 2018.

[2] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *Adv. Neural Inform. Process. Syst.*, 2017.

[3] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. First order motion model for image animation. In *NeurIPS*, December 2019.

[4] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (SIGGRAPH)*, July 2022.

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Adv. Neural Inform. Process. Syst.*, pages 2672–2680, 2014.

[6] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 2019.

[7] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017.

[8] Qi Shan, Riley Adams, Brian Curless, Yasutaka Furukawa, and Steven M Seitz. The Visual Turing Test for scene reconstruction. In *Proc. 3DV*, 2013.

[9] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images, 2020.

[10] Erroll Wood, Tadas Baltrušaitis, Charlie Hewitt, Sebastian Dziadzio, Thomas J Cashman, and Jamie Shotton. Fake it till you make it: Face analysis in the wild using synthetic data alone. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3681–3691, 2021.

[11] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *Int. Conf. Learn. Represent.*, 2018.

[12] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *Int. Conf. Learn. Represent.*, 2019.

[13] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[14] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 8110–8119, 2020.

[15] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. *Advances in Neural Information Processing Systems*, 34, 2021.

[16] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2009.

[17] Roger Mohr, Long Quan, and Françoise Veillon. Relative 3d reconstruction using multiple uncalibrated images. *Int. J. Robotics Research*, 14(6):619–632, 1995.

[18] Paul Beardsley, Phil Torr, and Andrew Zisserman. 3d model acquisition from extended image sequences. In *Eur. Conf. Comput. Vis.*, pages 683–695. Springer, 1996.

[19] Marc Pollefeys, Luc Van Gool, Maarten Vergauwen, Frank Verbiest, Kurt Cornelis, Jan Tops, and Reinhard Koch. Visual modeling with a hand-held camera. *Int. J. Comput. Vis.*, 59(3):207–232, 2004.

[20] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2010.

[21] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *Eur. Conf. Comput. Vis.*, 2016.

[22] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016.

[23] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *Eur. Conf. Comput. Vis.*, pages 649–666. Springer, 2016.

[24] Richard Zhang, Jun-Yan Zhu, Phillip Isola, Xinyang Geng, Angela S Lin, Tianhe Yu, and Alexei A Efros. Real-time user-guided image colorization with learned deep priors. *ACM Trans. Graph.*, 36(4):119, 2017.

[25] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew P Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017.

[26] Jingwen Chen, Jiawei Chen, Hongyang Chao, and Ming Yang. Image blind denoising with generative adversarial network based noise modeling. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3155–3164, 2018.

[27] Ricardo Martin-Brualla, Rohit Pandey, Shuoran Yang, Pavel Pidlypenskyi, Jonathan Taylor, Julien Valentin, Sameh Khamis, Philip Davidson, Anastasia Tkach, Peter Lincoln, Adarsh Kowdle, Christoph Rhemann, Dan B Goldman, Cem Keskin, Steve Seitz, Shahram Izadi, and Sean Fanello. LookinGood: Enhancing performance capture with real-time neural re-rendering. In *Proc. SIGGRAPH Asia*, 2018.

[28] Moustafa Meshry, Dan B Goldman, Sameh Khamis, Hugues Hoppe, Rohit Pandey, Noah Snavely, and Ricardo Martin-Brualla. Neural rerendering in the wild. *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[29] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM Trans. Graph.*, 2019.

[30] Hao Dong, Simiao Yu, Chao Wu, and Yike Guo. Semantic image synthesis via adversarial learning. In *Int. Conf. Comput. Vis.*, 2017.

[31] Xiaolong Wang and Abhinav Gupta. Generative image modeling using style and structure adversarial networks. In *Eur. Conf. Comput. Vis.*, 2016.

[32] Zhifei Zhang, Yang Song, and Hairong Qi. Age progression/regression by conditional adversarial autoencoder. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017.

[33] Thomas Gerig, Andreas Morel-Forster, Clemens Blumer, Bernhard Egger, Marcel Luthi, Sandro Schönborn, and Thomas Vetter. Morphable face models-an open framework. In *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*, pages 75–82. IEEE, 2018.

[34] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *Proc. SIGGRAPH Asia*, 34(6):248:1–248:16, October 2015.

[35] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *Int. Conf. Learn. Represent.*, 2016.

[36] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Adv. Neural Inform. Process. Syst.*, pages 5767–5777, 2017.

[37] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[38] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3D reconstruction in function space. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[39] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019.

[40] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. *arXiv preprint arXiv:2003.08934*, 2020.

[41] Vincent Sitzmann, Julien N.P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *Proc. NeurIPS*, 2020.

[42] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *Eur. Conf. Comput. Vis.*, 2018.

[43] Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Kumar Singh, and Ming-Hsuan Yang. Diverse image-to-image translation via disentangled representations. In *Eur. Conf. Comput. Vis.*, 2018.

[44] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *Int. Conf. Comput. Vis.*, pages 1511–1520, 2017.

[45] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.

[46] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Adv. Neural Inform. Process. Syst.*, pages 3483–3491, 2015.

[47] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *Int. Conf. Learn. Represent.*, 2016.

[48] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. *Int. Conf. Learn. Represent.*, 2016.

[49] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Eur. Conf. Comput. Vis.*, pages 694–711. Springer, 2016.

[50] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *neurips*, 25:1097–1105, 2012.

[51] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. In *Eur. Conf. Comput. Vis.*, pages 181–196, 2018.

[52] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Int. Conf. Comput. Vis.*, pages 843–852, 2017.

[53] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Int. Conf. Comput. Vis.*, pages 1422–1430, 2015.

[54] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *Eur. Conf. Comput. Vis.*, pages 69–84. Springer, 2016.

[55] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *Int. Conf. Learn. Represent.*, 2014.

[56] Animesh Karnewar and Raghu Sesha Iyengar. Msg-gan: Multi-scale gradients gan for more stable and synchronized multi-scale image synthesis. *arXiv preprint arXiv:1903.06048*, 2019.

[57] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Int. Conf. Comput. Vis.*, 2017.

[58] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. *ICML*, 2016.

[59] Mihaela Rosca, Balaji Lakshminarayanan, and David Warde-Farley Shakir Mohamed. Variational approaches for auto-encoding generative adversarial networks. *stat*, 1050:15, 2017.

[60] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[61] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *ICML*, pages 2642–2651. JMLR. org, 2017.

[62] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. *Adv. Neural Inform. Process. Syst.*, 29, 2016.

[63] Elman Mansimov, Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Generating images from captions with attention. *Int. Conf. Learn. Represent.*, 2016.

[64] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. *ICML*, 2016.

[65] Yedid Hoshen and Lior Wolf. Nam: Non-adversarial unsupervised domain mapping. In *Eur. Conf. Comput. Vis.*, pages 436–451, 2018.

[66] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. In *ICML*, 2017.

[67] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *Adv. Neural Inform. Process. Syst.*, 2017.

[68] Liqian Ma, Xu Jia, Stamatios Georgoulis, Tinne Tuytelaars, and Luc Van Gool. Exemplar guided unsupervised image-to-image translation. *Int. Conf. Learn. Represent.*, 2019.

[69] Amélie Royer, Konstantinos Bousmalis, Stephan Gouws, Fred Bertsch, Inbar Mosseri, Forrester Cole, and Kevin Murphy. Xgan: Unsupervised image-to-image translation for many-to-many mappings. *arXiv preprint arXiv:1711.05139*, 2017.

[70] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Int. Conf. Comput. Vis.*, pages 2223–2232, 2017.

[71] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 8789–8797, 2018.

[72] Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A Efros. Everybody dance now. *arXiv preprint arXiv:1808.07371*, 2018.

[73] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Nikolai Yakovenko, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. In *Adv. Neural Inform. Process. Syst.*, 2018.

[74] Amjad Almahairi, Sai Rajeshwar, Alessandro Sordoni, Philip Bachman, and Aaron Courville. Augmented CycleGAN: Learning many-to-many mappings from unpaired data. In *ICML*, 2018.

[75] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016.

[76] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 815–823, 2015.

[77] Yedid Hoshen, Ke Li, and Jitendra Malik. Non-adversarial image synthesis with generative latent nearest neighbors. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5811–5819, 2019.

[78] Ke Li and Jitendra Malik. Implicit maximum likelihood estimation. *arXiv preprint arXiv:1809.09087*, 2018.

[79] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, 2014.

[80] Ricardo Martin-Brualla. Space needle timelapse. `https://bit.ly/2RMKZ3B`, 2007. Accessed: 2019-09-22.

[81] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. *Brit. Mach. Vis. Conf.*, 2015.

[82] Wu Jie. Facial expression recognition. `https://github.com/WuJie1010/Facial-Expression-Recognition.Pytorch`, 2018. Accessed: 2019-09-22.

[83] KDEF. Karolinska directed emotional faces (kdef) dataset. `http://kdef.se`, 2017. Accessed: 2019-09-22.

[84] Sameer Agarwal, Noah Snavely, Ian Simon, Steven M Seitz, and Richard Szeliski. Building Rome in a day. In *Int. Conf. Comput. Vis.*, 2009.

[85] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo Tourism: Exploring photo collections in 3D. In *Proc. SIGGRAPH*, 2006.

[86] Eunbyung Park, Jimei Yang, Ersin Yumer, Duygu Ceylan, and Alexander C Berg. Transformation-grounded image generation network for novel 3D view synthesis. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017.

[87] Helge Rhodin, Mathieu Salzmann, and Pascal Fua. Unsupervised geometry-aware representation learning for 3D human pose estimation. In *Eur. Conf. Comput. Vis.*, 2018.

[88] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Multi-view 3D models from single images with a convolutional network. In *Eur. Conf. Comput. Vis.*, 2016.

[89] Takafumi Saito and Tokiichiro Takahashi. Comprehensible rendering of 3-D shapes. In *Proc. SIGGRAPH*, 1990.

[90] Hyeongwoo Kim, Pablo Garrido, Ayush Tewari, Weipeng Xu, Justus Thies, Matthias Niessner, Patrick Pérez, Christian Richardt, Michael Zollhöfer, and Christian Theobalt. Deep video portraits. In *Proc. SIGGRAPH*, 2018.

[91] Vu Hoang Hiep, Renaud Keriven, Patrick Labatut, and Jean-Philippe Pons. Towards high-resolution large-scale multi-view stereo. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2009.

[92] Paul Debevec, Camillo Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, pages 11–20, 1996.

[93] Steven Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael Cohen. The Lumigraph. In *SIGGRAPH'96: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*. ACM Press, New York, NY, USA, 1996.

[94] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. In *Proc. SIGGRAPH*, 2018.

[95] Rahul Garg, Hao Du, Steven M. Seitz, and Noah Snavely. The dimensionality of scene appearance. In *Int. Conf. Comput. Vis.*, 2009.

[96] Daniel Hauagge, Scott Wehrwein, Paul Upchurch, Kavita Bala, and Noah Snavely. Reasoning about photo collections using models of outdoor illumination. In *Brit. Mach. Vis. Conf.*, 2014.

[97] Pierre-Yves Laffont, Adrien Bousseau, Sylvain Paris, Frédo Durand, and George Drettakis. Coherent intrinsic images from photo collections. In *Proc. SIGGRAPH Asia*, 2012.

[98] Filip Radenović, Johannes Lutz Schönberger, Dinghuang Ji, Jan-Michael Frahm, Ondrej Chum, and Jiri Matas. From dusk till dawn: Modeling in the dark. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016.

[99] Kalyan Sunkavalli, Wojciech Matusik, Hanspeter Pfister, and Szymon Rusinkiewicz. Factored time-lapse video. In *Proc. SIGGRAPH*, 2007.

[100] Pierre-Yves Laffont, Zhile Ren, Xiaofeng Tao, Chao Qian, and James Hays. Transient attributes for high-level understanding and editing of outdoor scenes. In *Proc. SIGGRAPH*, 2014.

[101] Yannick Hold-Geoffroy, Kalyan Sunkavalli, Sunil Hadap, Emiliano Gambaretto, and Jean-François Lalonde. Deep outdoor illumination estimation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017.

[102] Tuanfeng Y. Wang, Tobias Ritschel, and Niloy J. Mitra. Joint material and illumination estimation from photo sets in the wild. In *Proc. 3DV*, 2018.

[103] SM Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S Morcos, Marta Garnelo, Avraham Ruderman, Andrei A Rusu, Ivo Danihelka, Karol Gregor, et al. Neural scene representation and rendering. *Science*, 2018.

[104] Johannes L Schönberger. Colmap. `http://colmap.github.io`, 2016.

[105] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016.

[106] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proc. Eurographics Symposium on Geometry Processing*, 2006.

[107] Michael Waechter, Nils Moehrle, and Michael Goesele. Let there be color! Large-scale texturing of 3D reconstructions. In *Eur. Conf. Comput. Vis.*, 2014.

[108] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2018.

[109] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ADE20K dataset. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017.

[110] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016.

[111] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks. *Google Research Blog. Retrieved June*, 2015.

[112] Egor Zakharov, Aliaksandra Shysheya, Egor Burkov, and Victor Lempitsky. Few-shot adversarial learning of realistic neural talking head models. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 9459–9468, 2019.

[113] Egor Burkov, Igor Pasechnik, Artur Grigorev, and Victor Lempitsky. Neural head reenactment with latent pose descriptors. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 13786–13795, 2020.

[114] Egor Zakharov, Aleksei Ivakhnenko, Aliaksandra Shysheya, and Victor Lempitsky. Fast bi-layer neural synthesis of one-shot realistic head avatars. In *Eur. Conf. Comput. Vis.*, August 2020.

[115] Justus Thies, Michael Zollhöfer, Matthias Nießner, Levi Valgaerts, Marc Stamminger, and Christian Theobalt. Real-time expression transfer for facial reenactment. *ACM Trans. Graph.*, 34(6):183–1, 2015.

[116] Justus Thies, Michael Zollhofer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. Face2face: Real-time face capture and reenactment of rgb videos. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2387–2395, 2016.

[117] Supasorn Suwajanakorn, Steven M Seitz, and Ira Kemelmacher-Shlizerman. Synthesizing obama: learning lip sync from audio. *ACM Trans. Graph.*, 36(4):1–13, 2017.

[118] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *Proc. SIGGRAPH*, pages 187–194, 1999.

[119] Guy Gafni, Justus Thies, Michael Zollhöfer, and Matthias Nießner. Dynamic neural radiance fields for monocular 4d facial avatar reconstruction. *arXiv preprint arXiv:2012.03065*, 2020.

[120] Koki Nagano, Jaewoo Seo, Jun Xing, Lingyu Wei, Zimo Li, Shunsuke Saito, Aviral Agarwal, Jens Fursund, and Hao Li. pagan: real-time avatars using dynamic textures. *ACM Trans. Graph.*, 37(6):1–12, 2018.

[121] Ohad Fried, Ayush Tewari, Michael Zollhöfer, Adam Finkelstein, Eli Shechtman, Dan B Goldman, Kyle Genova, Zeyu Jin, Christian Theobalt, and Maneesh Agrawala. Text-based editing of talking-head video. *ACM Trans. Graph.*, 38(4):1–14, 2019.

[122] Lele Chen, Ross K Maddox, Zhiyao Duan, and Chenliang Xu. Hierarchical cross-modal talking face generation with dynamic pixel-wise loss. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 7832–7841, 2019.

[123] Yuval Nirkin, Yosi Keller, and Tal Hassner. Fsgan: Subject agnostic face swapping and reenactment. In *Int. Conf. Comput. Vis.*, pages 7184–7193, 2019.

[124] Albert Pumarola, Antonio Agudo, Aleix M Martinez, Alberto Sanfeliu, and Francesc Moreno-Noguer. Ganimation: Anatomically-aware facial animation from a single image. In *Eur. Conf. Comput. Vis.*, pages 818–833, 2018.

[125] Kuangxiao Gu, Yuqian Zhou, and Thomas Huang. Flnet: Landmark driven fetching and learning network for faithful talking facial animation synthesis. In *AAAI*, volume 34, pages 10861–10868, 2020.

[126] Sungjoo Ha, Martin Kersner, Beomsu Kim, Seokjun Seo, and Dongyoung Kim. Marionette: Few-shot face reenactment preserving identity of unseen targets. In *AAAI*, volume 34, pages 10893–10900, 2020.

[127] Ting-Chun Wang, Ming-Yu Liu, Andrew Tao, Guilin Liu, Jan Kautz, and Bryan Catanzaro. Few-shot video-to-video synthesis. In *NeurIPS*, 2019.

[128] Aayush Bansal, Shugao Ma, Deva Ramanan, and Yaser Sheikh. Recycle-gan: Unsupervised video retargeting. In *Eur. Conf. Comput. Vis.*, pages 119–135, 2018.

[129] Wayne Wu, Yunxuan Zhang, Cheng Li, Chen Qian, and Chen Change Loy. Reenactgan: Learning to reenact faces via boundary transfer. In *Eur. Conf. Comput. Vis.*, pages 603–619, 2018.

[130] Olivia Wiles, A Koepke, and Andrew Zisserman. X2face: A network for controlling face generation using images, audio, and pose codes. In *Eur. Conf. Comput. Vis.*, pages 670–686, 2018.

[131] Ting-Chun Wang, Arun Mallya, and Ming-Yu Liu. One-shot free-view neural talking-head synthesis for video conferencing. *arXiv preprint arXiv:2011.15126*, 2020.

[132] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Int. Conf. Comput. Vis.*, pages 1501–1510, 2017.

[133] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[134] Xihui Liu, Guojun Yin, Jing Shao, Xiaogang Wang, and Hongsheng Li. Learning to predict layout-to-image conditional convolutions for semantic image synthesis. In *NeurIPS*, 2019.

[135] Peihao Zhu, Rameen Abdal, Yipeng Qin, and Peter Wonka. Sean: Image synthesis with semantic region-adaptive normalization. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5104–5113, 2020.

[136] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[137] Cheng-Han Lee, Ziwei Liu, Lingyun Wu, and Ping Luo. Maskgan: Towards diverse and interactive facial image manipulation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[138] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *International conference on machine learning*, pages 3481–3490. PMLR, 2018.

[139] J. S. Chung, A. Nagrani, and A. Zisserman. VoxCeleb2: Deep Speaker Recognition. In *INTERSPEECH*, 2018.

[140] Adrian Bulat and Georgios Tzimiropoulos. How far are we from solving the 2d & 3d face alignment problem?(and a dataset of 230,000 3d facial landmarks). In *Int. Conf. Comput. Vis.*, pages 1021–1030, 2017.

[141] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[142] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *neurips*, 2017.

[143] Ruben Tolosana, Ruben Vera-Rodriguez, Julian Fierrez, Aythami Morales, and Javier Ortega-Garcia. Deepfakes and beyond: A survey of face manipulation and fake detection. *Information Fusion*, 64:131–148, 2020.

[144] Yisroel Mirsky and Wenke Lee. The creation and detection of deepfakes: A survey. *ACM Computing Surveys (CSUR)*, 54(1):1–41, 2021.

[145] Francesco Marra, Diego Gragnaniello, Davide Cozzolino, and Luisa Verdoliva. Detection of gan-generated fake images over social networks. In *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 384–389. IEEE, 2018.

[146] Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A Efros. Cnn-generated images are surprisingly easy to spot... for now. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8695–8704, 2020.

[147] Xu Zhang, Svebor Karaman, and Shih-Fu Chang. Detecting and simulating artifacts in gan fake images. In *2019 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6. IEEE, 2019.

[148] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Int. Conf. Learn. Represent.*, 2015.

[149] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields. *arXiv preprint arXiv:2103.13415*, 2021.

[150] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the wild: Neural radiance fields for unconstrained photo collections. In *CVPR*, 2021.

[151] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *NeurIPS*, 2020.

[152] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H. Mueller, Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, and Markus Steinberger. DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks. *Computer Graphics Forum*, 40(4), 2021.

[153] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B. Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. *ICCV*, 2021.

[154] Ayush Tewari, Justus Thies, Ohad Fried, Vincent Sitzmann, Kalyan Sunkavalli, Ricardo Martin-Brualla, Tomas Simon, Jason Saragih, Matthias Nießner, Rohit K. Pandey, Sean Fanello, Gordon Wetzstein, Jun-Yan Zhu, Christian Theobalt, Maneesh Agrawala, Eli Shechtman, Dan B. Goldman, and Michael Zollhöfer. Advances in neural rendering. In *ACM SIGGRAPH 2021 Courses*, pages 1–320, 2021.

[155] Michael Oechsle, Songyou Peng, and Andreas Geiger. UNISURF: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *International Conference on Computer Vision (ICCV)*, 2021.

[156] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *NeurIPS*, 2021.

[157] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural geometric level of detail: Real-time rendering with implicit 3D shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[158] Yann Labbé, Justin Carpentier, Mathieu Aubry, and Josef Sivic. CosyPose: Consistent multi-view multi-object 6D pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[159] Erroll Wood, Tadas Baltrušaitis, Charlie Hewitt, Sebastian Dziadzio, Matthew Johnson, Virginia Estellers, Thomas J. Cashman, and Jamie Shotton. Fake it till you make it: Face analysis in the wild using synthetic data alone. In *ICCV*, 2021.

[160] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Häusser, Canaer Hazırbaş, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning optical flow with convolutional networks. In *ICCV*, 2015.

[161] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *CVPR Workshop on Autonomous Driving (WAD)*, 2018.

[162] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. In *CoRL*, 2018.

[163] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4), 2017.

[164] Yao Yao, Zixin Luo, Shiwei Li, Jingyang Zhang, Yufan Ren, Lei Zhou, Tian Fang, and Long Quan. BlendedMVS: A large-scale dataset for generalized multi-view stereo networks. *Computer Vision and Pattern Recognition (CVPR)*, 2020.

[165] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 406–413, 2014.

[166] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3D-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems*, 2019.

[167] Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common objects in 3D: Large-scale learning and evaluation of real-life 3D category reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10901–10911, 2021.

[168] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured Lumigraph rendering. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, pages 425–432, 2001.

[169] Sudipta Sinha, Drew Steedly, and Rick Szeliski. Piecewise planar stereo for image-based rendering. In *2009 International Conference on Computer Vision*, 2009.

[170] Gaurav Chaurasia, Olga Sorkine, and George Drettakis. Silhouette-aware warping for image-based rendering. *Computer Graphics Forum*, 30(4):1223–1232, 2011.

[171] Gaurav Chaurasia, Sylvain Duchene, Olga Sorkine-Hornung, and George Drettakis. Depth synthesis and local warps for plausible image-based navigation. *ACM Transactions on Graphics (TOG)*, 32(3):1–12, 2013.

[172] Eric Penner and Li Zhang. Soft 3D reconstruction for view synthesis. *ACM Transactions on Graphics (TOG)*, 36(6):1–11, 2017.

[173] Gernot Riegler and Vladlen Koltun. Free view synthesis. In *European Conference on Computer Vision*, pages 623–640. Springer, 2020.

[174] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. *NeurIPS*, 2017.

[175] Shubham Tulsiani, Tinghui Zhou, Alexei A. Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2626–2634, 2017.

[176] Philipp Henzler, Niloy Mitra, and Tobias Ritschel. Learning a neural 3D texture space from 2D exemplars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8356–8364, 2020.

[177] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM Trans. Graph.*, 38(4):65:1–65:14, July 2019.

[178] Inchang Choi, Orazio Gallo, Alejandro Troccoli, Min H. Kim, and Jan Kautz. Extreme view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7781–7790, 2019.

[179] Pratul P. Srinivasan, Richard Tucker, Jonathan T. Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. Pushing the boundaries of view extrapolation with multiplane images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 175–184, 2019.

[180] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 37, 2018.

[181] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. Deepview: View synthesis with learned gradient descent. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2367–2376, 2019.

[182] Zhengqi Li, Wenqi Xian, Abe Davis, and Noah Snavely. Crowdsampling the plenoptic function. In *European Conference on Computer Vision*, pages 178–196. Springer, 2020.

[183] Frank Dellaert and Lin Yen-Chen. Neural volume rendering: NeRF and beyond. *arXiv preprint arXiv:2101.05204*, 2020.

[184] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. NeRF++: Analyzing and improving neural radiance fields. *arXiv:2010.07492*, 2020.

[185] Daniel Rebain, Wei Jiang, Soroosh Yazdani, Ke Li, Kwang Moo Yi, and Andrea Tagliasacchi. DeRF: Decomposed radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14153–14161, 2021.

[186] Matthew Tancik, Ben Mildenhall, Terrance Wang, Divi Schmidt, Pratul P. Srinivasan, Jonathan T. Barron, and Ren Ng. Learned initializations for optimizing coordinate-based neural representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2846–2855, 2021.

[187] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. Space-time neural irradiance fields for free-viewpoint video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9421–9431, 2021.

[188] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6498–6508, 2021.

[189] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural Radiance Fields for Dynamic Scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[190] Yilun Du, Yinan Zhang, Hong-Xing Yu, Joshua B. Tenenbaum, and Jiajun Wu. Neural radiance flow for 4D view synthesis and video processing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

[191] Guy Gafni, Justus Thies, Michael Zollhöfer, and Matthias Nießner. Dynamic neural radiance fields for monocular 4D facial avatar reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8649–8658, June 2021.

[192] Chen Gao, Yichang Shih, Wei-Sheng Lai, Chia-Kai Liang, and Jia-Bin Huang. Portrait neural radiance fields from a single image. *arXiv preprint arXiv:2012.05903*, 2020.

[193] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. GRAF: Generative radiance fields for 3D-aware image synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[194] Eric Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-GAN: Periodic implicit generative adversarial networks for 3D-aware image synthesis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[195] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. PlenOctrees for real-time rendering of neural radiance fields. In *ICCV*, 2021.

[196] Alex Trevithick and Bo Yang. GRF: Learning a general radiance field for 3D scene representation and rendering. In *arXiv:2010.04595*, 2020.

[197] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P. Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *CVPR*, 2021.

[198] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. MVSNeRF: Fast generalizable radiance field reconstruction from multi-view stereo, 2021.

[199] Jonathan Tremblay, Thang To, and Stan Birchfield. Falling things: A synthetic dataset for 3D object detection and pose estimation. In *CVPR Workshop on Real World Challenges and New Benchmarks for Deep Learning in Robotic Vision*, 2018.

[200] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio Lopez. The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *CVPR*, 2016.

[201] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *CVPR*, 2016.

[202] Ankur Handa, Viorica Pătrăucean, Vijay Badrinarayanan, Simon Stent, and Roberto Cipolla. SceneNet: Understanding real world indoor scenes with synthetic data. In *arXiv 1511.07041*, 2015.

[203] Yi Zhang, Weichao Qiu, Qi Chen, Xaolin Hu, and Alan Yuille. UnrealStereo: A synthetic dataset for analyzing stereo vision. In *arXiv:1612.04647*, 2016.

[204] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *European Conference on Computer Vision (ECCV)*, pages 102–118, 2016.

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

[206] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An interactive 3D environment for visual AI. *arXiv preprint arXiv:1712.05474*, 2017.

[207] Thang To, Jonathan Tremblay, Duncan McKay, Yukie Yamaguchi, Kirby Leung, Adrian Balanon, Jia Cheng, and Stan Birchfield. NDDS: NVIDIA deep learning dataset synthesizer, 2018. `https://github.com/NVIDIA/Dataset_Synthesizer`.

[208] Adam Crespi, Cesar Romero, Srinivas Annambhotla, Jonathan Hogins, and Alex Thaman. Unity perception, 2020. `https://blogs.unity3d.com/2020/06/10/`.

[209] Maximilian Denninger, Martin Sundermeyer, Dominik Winkelbauer, Youssef Zidan, Dmitry Olefir, Mohamad Elbadrawy, Ahsan Lodhi, and Harinandan Katam. BlenderProc. *arXiv preprint arXiv:1911.01911*, 2019.

[210] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. SAPIEN: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11097–11107, 2020.

[211] Jinwei Gu, Xiaodong Yang, Shalini De Mello, and Jan Kautz. Dynamic facial analysis: From Bayesian filtering to recurrent neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[212] Bernardo Iraci. *Blender Cycles: Lighting and Rendering Cookbook*. Packt Publishing Ltd, 2013.

[213] Shreeyak S. Sajjan, Matthew Moore, Mike Pan, Ganesh Nagaraja, Johnny Lee, Andy Zeng, and Shuran Song. ClearGrasp: 3D shape estimation of transparent objects for manipulation. *arXiv preprint arXiv:1910.02550*, 2019.

[214] Erroll Wood, Tadas Baltrusaitis, Xucong Zhang, Yusuke Sugano, Peter Robinson, and Andreas Bulling. Rendering of eyes for eye-shape registration and gaze estimation. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

[215] Roey Ron and Gil Elbaz. EXPO-HD: Exact object perception using high distraction synthetic data. In *arXiv:2007.14354*, 2020.

[216] Christoph Heindl, Sebastian Zambal, and Josef Scharinger. Learning to predict robot keypoints using artificially generated images. In *24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1536–1539, 2019.

[217] Steven Parker, James Bigler, Andreas Dietrich, Heiko Friedrich, Jared Hoberock, David Luebke, David McAllister, Morgan McGuire, Keith Morley, and Austin Robison. OptiX: A General Purpose Ray Tracing Engine. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 2010.

[218] Nathan Morrical, Jonathan Tremblay, Yunzhi Lin, Stephen Tyree, Stan Birchfield, Valerio Pascucci, and Ingo Wald. NViSII: A scriptable tool for photorealistic image generation. In *ICLR Workshop on Synthetic Data Generation*, May 2021.

[219] Jeffrey Ichnowski, Yahav Avigal, Justin Kerr, and Ken Goldberg. Dex-NeRF: Using a neural radiance field to grasp transparent objects. In *Conference on Robot Learning (CoRL)*, 2020.

[220] Jun Yang, Yizhou Gao, Dong Li, and Steven L Waslander. Robi: A multi-view dataset for reflective objects in robotic bin-picking. *arXiv preprint arXiv:2105.04112*, 2021.

[221] Philipp Henzler, Jeremy Reizenstein, Patrick Labatut, Roman Shapovalov, Tobias Ritschel, Andrea Vedaldi, and David Novotny. Unsupervised learning of 3D object categories from videos in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4700–4709, 2021.

[222] Brent Burley. Extending the Disney BRDF to a BSDF with integrated subsurface scattering. *Physically Based Shading in Theory and Practice SIGGRAPH Course*, 2015.

[223] GoogleResearch. Google Scanned Objects. In *Open Robotics*, 2021.

[224] Erwin Coumans and Yunfei Bai. PyBullet, a Python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2019.

[225] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. ABC: A big CAD model dataset for geometric deep learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[226] Jasmine Collins, Shubham Goel, Achleshwar Luthra, Leon Xu, Kenan Deng, Xi Zhang, Tomas F Yago Vicente, Himanshu Arora, Thomas Dideriksen, Matthieu Guillaumin, and Jitendra Malik. Abo: Dataset and benchmarks for real-world 3d object understanding. *arXiv preprint arXiv:2110.06199*, 2021.

[227] Jan Novák, Iliyan Georgiev, Johannes Hanika, and Wojciech Jarosz. Monte Carlo methods for volumetric light transport simulation. *Computer Graphics Forum*, 37(2):551–576, 2018.

[228] Alexander Majercik, Cyril Crassin, Peter Shirley, and Morgan McGuire. A ray-box intersection algorithm and efficient dynamic voxel rendering. *Journal of Computer Graphics Techniques Vol*, 7(3):66–81, 2018.

[229] Duane Merrill. CUB: A library of warp-wide, block-wide, and device-wide GPU parallel primitives, 2017. `https://nvlabs.github.io/cub/`.

[230] Krishna Murthy Jatavallabhula, Edward Smith, Jean-Francois Lafleche, Clement Fuji Tsang, Artem Rozantsev, Wenzheng Chen, Tommy Xiang, Rev Lebaredian, and Sanja Fidler. Kaolin: A PyTorch library for accelerating 3D deep learning research. *arXiv:1911.05063*, 2019.

[231] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[232] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[233] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Adv. Neural Inform. Process. Syst.*, 34, 2021.

[234] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *arXiv preprint arXiv:2112.10752*, 2021.

[235] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022.

[236] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2022.

[237] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 12873–12883, 2021.

[238] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *ICML*, pages 1691–1703. PMLR, 2020.

[239] Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *ICML*, pages 1747–1756. PMLR, 2016.

[240] Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. Hologan: Unsupervised learning of 3d representations from natural images. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 7588–7597, 2019.

[241] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[242] Jiatao Gu, Lingjie Liu, Peng Wang, and Christian Theobalt. StyleNeRF: A style-based 3d aware generator for high-resolution image synthesis. In *International Conference on Learning Representations (ICLR)*, 2022.

[243] Roy Or-El, Xuan Luo, Mengyi Shan, Eli Shechtman, Jeong Joon Park, and Ira Kemelmacher-Shlizerman. StyleSDF: High-Resolution 3D-Consistent Image and Geometry Generation. 2022.

[244] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3D generative adversarial networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[245] Ming-Yu Liu. GauGAN2. `http://gaugan.org/gaugan2/`, 2021. Accessed: 2022-05-22.

[246] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[247] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. A general differentiable mesh renderer for image-based 3d reasoning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[248] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.