Learning View Synthesis from Minimal Scene Specifications

Thesis approved by
the Department of Computer Science
RPTU University Kaiserslautern-Landau
for the award of the Doctoral Degree
Doctor of Engineering (Dr.-Ing.)

to

Tewodros Amberbir Habtegebrial

Date of Defense: 11.04.2025

Dean:	Prof. Dr. Christoph Garth
Chair:	Prof. Dr. Christoph Grimm
Reviewer:	Prof. Dr. Didier Stricker
Reviewer:	Prof. Dr. Yiorgos Chrysanthou
Reviewer:	Dr. Fabio Carrara

DE-386

Abstract

Computer-generated images, videos, and visual effects are indispensable resources for digital content creation that enable artists to create engaging visual stories. However, creating compelling stories often requires photorealistic 3D models that are designed by skilled digital artists. Novel View Synthesis (NVS) has emerged as a cheaper means to achieve photorealism. NVS uses a collection of photographs to render a scene from novel camera poses without relying on expensive models of geometry, materials, or light.

This thesis introduces a number of view synthesis approaches with varying levels of input complexity, ranging from multi-view stereo and sparse multi-view spherical images to 2D semantic maps and simple textual descriptions. The proposed methods improve the state-of-the-art in terms of accuracy, efficiency, and usability. In the multi-view stereo input setting, the proposed FastNVS method significantly outperforms existing techniques in terms of speed and accuracy. FastNVS achieves this by decomposing the NVS problem into two structured prediction tasks, namely, proxy geometry estimation and texture inpainting. In the more challenging setting of spherical input images, prior work relies on the Multisphere Images (MSI) scene representation. MSI-based methods achieve fast rendering speed but are limited to modeling low-dimensional color values per-sphere. To alleviate this, we propose a novel scene representation called Soft Occlusion Multisphere Images (SOMSI) that enables modeling high-dimensional appearance features in MSI. This is achieved by assigning appearance features to a few occlusion levels, instead of a large number of MSI spheres. SOMSI produces novel views with significantly higher quality while retaining the fast rendering times of traditional MSI.

Furthermore, the usability of view synthesis methods is enhanced by introducing novel techniques that require minimal user input and grant users control over the 3D scene. Additionally, the proposed methods give users great creative freedom by enabling them to create novel 3D scenes for which no input images exist. The first method in this line of research is the GVSNet technique, which allows users to create novel views from a single input 2D semantic sketch. Users can also manipulate the geometry of existing scenes by editing the input semantic map. However, the process of creating and editing a semantic map can easily become a tedious task. In order to further simplify the creative process, a novel approach is proposed that maps input textual scene descriptions into renderable scene representations. The proposed method, Text2MPI, is a diffusion model trained to generate compact Multiplane Images (MPI) representations from text. Text2MPI generates crisp photorealistic novel views that are 3D-consistent and match the input text description. Furthermore, the proposed model harnesses the vast generalization capability of 2D diffusion models by integrating 2D scene priors into its training procedure.

Acknowledgment

First, I would like to express my sincere gratitude to my supervisor, Prof. Didier Stricker, for his unwavering guidance. I am forever grateful for the trust he placed in me and for giving me the space and encouragement to aim higher. I would also like to thank the members of my PhD committee for their valuable feedback. I am also grateful to the Computer Science Department at RPTU and to the Google PhD Fellowship Program for their support of my research.

My colleagues at DFKI and RPTU provided a wonderfully supportive and collaborative environment. I thank Gerd, Christiano, Mohamed, Christian, Stephan, Reza, Kiran, David, Federico, Alireza, Abdalla, Pragati, and Marcel for their help and camaraderie. Special thanks to Levy Michelly Kaul, who helped me navigate life in Germany during my first semester. I would also like to thank the administrative staff for their support, especially Mrs. Ownes, Keonna, Moni, and Eric for their assistance with administrative matters.

I am profoundly thankful for the support and collaborations within the research community. I would like to thank Varun Jampani, from whom I learned a great deal about formulating the right research questions and about the importance of rigorous empirical evaluation. My sincere thanks also go to David Salesin for his invaluable mentorship. I am also grateful to my internship mentor, Stefan Popov, whose technical brilliance and kindness I will always cherish. I am grateful to Firsum Reda for his brotherly advice and encouragement, and to Orazio Gallo for his technical mentorship.

I am also deeply indebted to my friends, both in Kaiserslautern and in Ethiopia. Thank you to Reza and Sandra for the wonderful times outside the office, and to Lias for providing much-needed distractions. The Ethiopian community in Kaiserslautern, including Dere, Tigi, Abraham, Dani, Birhanu, Hany, Susu, Mengisti, Felix, Amanuel, Selam, and Tsige, made me feel truly at home, and I am deeply grateful for their presence.

Finally, I would like to express my heartfelt gratitude to my parents, aunts, and uncles. I am especially thankful for the unwavering love and support of my grandmother.

Above all, I thank my beautiful wife for her patience, constant support, for believing in me when I needed it most, and for serving as a sounding board for my ideas throughout this journey.

Contents

1 Introduction **11**
- 1.1 Image-based Rendering: The World as a Photorealistic Rendering Engine 12
- 1.2 The Image-based Rendering Continuum 14
- 1.3 Thesis Overview 15
- 1.4 Thesis Organization and Contributions 19
- 1.5 Publications 20

2 Preliminaries **23**
- 2.1 Generative Models 24
  - 2.1.1 Generative Adversarial Networks 24
  - 2.1.2 Diffusion Models 28
- 2.2 Neural Rendering 31
  - 2.2.1 Multiplane Images 31
  - 2.2.2 Multisphere Images 33
  - 2.2.3 Neural Radiance Fields 36

3 Deep Stereo Vision Guided View Interpolation **41**
- 3.1 Introduction 42
- 3.2 Related Work 44
- 3.3 Proposed Method 45
  - 3.3.1 Depth Prediction Network 45
  - 3.3.2 Forward Mapping 50
  - 3.3.3 Texture Inpainting 51
- 3.4 Experiments 52
- 3.5 Conclusion and Future Work 57

4 Spherical Novel View Synthesis with Soft Occlusion Multisphere Images **59**
- 4.1 Introduction 60
- 4.2 Related Work 62
- 4.3 Preliminaries 63
- 4.4 Approach 64
  - 4.4.1 Soft Occlusion Multisphere Images (SOMSI) 64
  - 4.4.2 Modeling view dependent effects. 68
  - 4.4.3 Learning SOMSI with Coordinate MLP 69
- 4.5 Experiments 69

4.6 Network Architecture and Training Details . . . 76
4.6.1 Network Architecture . . . 76
4.7 Calibrating Spherical Light Fields through Perspective Cut-outs . . . 77
4.8 Conclusion . . . 78

5 Generative View Synthesis: from Single-view Semantics to Multi-view Images 79
5.1 Introduction . . . 80
5.2 Related Works . . . 82
5.3 Generative View Synthesis Model . . . 83
5.3.1 Semantics Uplifting Network . . . 84
5.3.2 Layered Translation Network . . . 84
5.3.3 Appearance Decoder Network . . . 85
5.4 Dataset, Experiments and Evaluations . . . 87
5.5 Conclusion . . . 95
5.6 Additional Experimental Details . . . 95
5.6.1 Dataset Details . . . 95
5.6.2 Training Losses . . . 96
5.6.3 Network Architecture . . . 96

6 Learning to Generate Multiplane Image Scene Representations from Text 101
6.1 Introduction . . . 102
6.2 Related Work . . . 104
6.3 Proposed Method . . . 105
6.3.1 Latent Multiplane Images . . . 105
6.3.2 Latent MPI Diffusion Model . . . 107
6.3.3 Generated 2D Prior . . . 108
6.3.4 Improving Scene Geometry and Extending Camera Motion . . . 110
6.3.5 Network Architectures . . . 111
6.4 Experiments . . . 111
6.5 Conclusion . . . 118

7 Conclusion 121
7.1 Thesis Summary . . . 122
7.2 Future Work . . . 123

List of Abbreviations 125

List of Figures 127

List of Tables 129

Bibliography **131**

1 Introduction

This chapter provides an overview of the thesis. It gives a brief introduction to the Image-based Rendering (IBR) paradigm, and it highlights the main research problems addressed in this thesis together with its main contributions. The organization of the rest of the thesis and a list of publications are also presented.

Contents

1.1 Image-based Rendering: The World as a Photorealistic Rendering Engine **12**

1.2 The Image-based Rendering Continuum **14**

1.3 Thesis Overview **15**

1.4 Thesis Organization and Contributions **19**

1.5 Publications **20**

The practice of storytelling is a universal tradition that is present across cultures and time periods, serving various purposes ranging from chronicling historical events to shaping cultural norms and moral values. Throughout the evolution of our civilization, storytelling took on diverse forms of communication. In contemporary society, cinema stands as one of the most compelling mediums of storytelling. Since the latter part of the 20th century, computer-generated imagery (CGI) has immensely enhanced the storytelling potential of cinema. CGI now enables simulating challenging camera movements, creation of powerful visual effects, and construction of immersive and dynamic worlds. Computer graphics tools are employed to render these virtual worlds while computer vision technologies like Dense 3D Reconstruction are also used to acquire detailed computer models of the real-world. Often, these real and virtual worlds are seamlessly integrated to create compelling cinematic stories.

Creating captivating virtual worlds requires highly detailed and accurate models of geometry, material properties, and lighting. In certain scenarios, non-photorealistic renderings may be suitable or even preferable. However, the objective in most cases is to generate photorealistic 3D scenes that are indistinguishable from reality. Significant advancements have been made in computer graphics research towards this goal. Nevertheless, achieving photorealism still requires a substantial investment of manual labor by highly skilled digital artists.

1.1 Image-based Rendering: The World as a Photorealistic Rendering Engine

The Image-based Rendering (IBR) paradigm emerged as a means to expedite the path towards photorealism by treating *the physical world and a camera* as a photorealistic rendering engine. IBR takes a set of input photographs of a real scene and aims to create new images of the scene, by rearranging the color pixels present in the input images. The input images are processed to create an explicit (or implicit) intermediate 3D representation which is then used to guide the computation of novel views. As the input images are already photographs of real-world scenes, photorealism is achieved almost at no cost in IBR.

IBR promises users the capability to freely navigate 3D scenes using only a set of input images. However, solving IBR requires addressing several ill-posed problems, such as estimating scene geometry and material properties from a sparse set of input images. IBR becomes particularly more challenging when the 3D scene involves thin geometric structures, and surfaces with non-diffuse, reflective, and refractive properties. The ill-posed nature of IBR has prompted prior works to use additional proxy scene geometry [21, 12, 29, 16] or learned scene priors [149, 136, 128].

Image-based Rendering, also referred to as Novel View Synthesis (NVS), has a rich history starting with seminal works such as View Interpolation [21], light fields [73] and Lumigraphs [43], extending up to some of the most exciting research works in computer vision today [83, 85, 68]. In the last few decades IBR has led to several creative applications

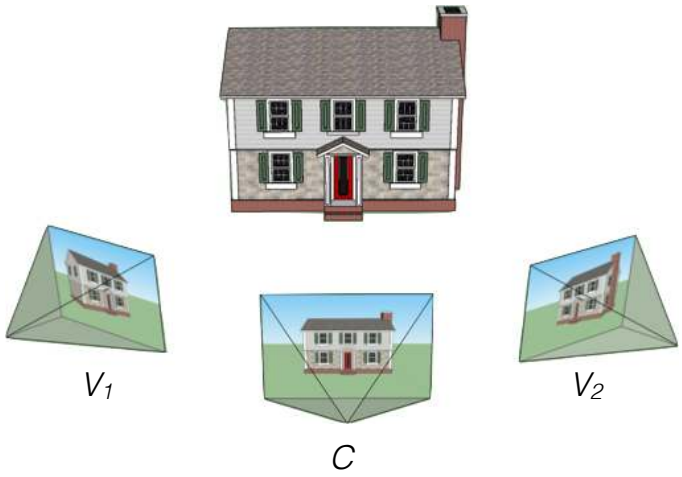


Figure 1.1: **Image-based Rendering.** Given input images V_1 and V_2, the goal of an IBR approach is to render the scene from novel camera viewpoints such as C. The input images and their camera parameters serve as a *scene specification*, similar to how a combination of meshes, texture-maps, material properties etc specify a scene in traditional graphics. Image source: Flynn *et al*. [36]

such as Photo-tourism [116], Microsoft's Photosynth and Free-viewpoint Video [152]. Recently, IBR tools are being used in the production of music videos [1].

Modern, neural IBR approaches (also referred to as neural rendering), offer robust end-to-end trainable solutions that require no external sources of proxy-scene geometry or brittle pre-processing steps. The success of neural approaches for IBR was spearheaded by the DeepStereo approach of Flynn *et al*. [36]. Zhou *et al*. [149] reintroduced the Multiplane Images (MPI) scene representation which was introduced in [121] in the context of stereo reconstruction. The advent of MPI-based view synthesis was a notable milestone that popularized learning-based view synthesis. MPIs enable affordable Novel View Synthesis models that can be trained using convolutional neural networks over datasets of posed multi-view images. Arguably, the most consequential moment in the recent history of view synthesis research is the introduction of Neural Radiance Fields (NeRF) [83]. NeRFs have led to unprecedented progress in IBR research and they have amassed substantial interest from the computer vision, graphics, and machine learning communities. NeRF-based approaches have been proposed for view synthesis of dynamic scenes [93], scenes with reflective surfaces [131], and view synthesis from internet photo collections [80]. NeRFs have also found applications in other tasks such as multi-view stereo reconstruction [18], long-range dense tracking [132], robotics [140], simultaneous localization and mapping [103], etc.

[1] https://youtu.be/918oGJFtjjw?t=134

1.2 The Image-based Rendering Continuum

The designs of Image-based Rendering techniques vary along several axes, such as dependence on explicit geometry versus input images, reliance on learned priors, amortized vs iterative optimization, input complexity, the degree of control they grant users, etc. Selecting the right IBR approach requires considering several constraints, such as the complexity of the desired scene (single objects or complex scenes with intricate geometry), the target rendering speed, and hardware limitations for training and inference. This section examines some continuums that are relevant to this thesis and modern NVS techniques. For a comprehensive review of classical and recent IBR approaches, refer to the excellent review papers: [82, 113, 114] and [138].

Scene Specification Complexity. Unlike traditional graphics pipelines, in IBR a set of input photographs and their camera parameters serve as a *scene specification*. Existing view synthesis works require varying number of input images. Methods such as light fields [73], Lumigraphs [43] require a large number of input structured images. In contrast, methods that rely on priors and generalization capabilities of deep neural networks [149, 35, 36] and proxy scene geometry [90, 16, 17] require fewer input images. At the extreme end of this spectrum, there are monocular view synthesis approaches that require only a single input image [128, 136, 24, 87] in order to perform Novel View Synthesis. However, methods that rely on few input images only permit rendering views that are in close proximity to the input image. This is due to the limited 3D volume captured by the viewing frustum of the input image [136, 24], the challenging nature of single-view 3D perception, and the Nyquist sampling limit imposed on the underlying discrete scene representation [128]. However, single-view NVS methods allow users to quickly create interesting visual effects such as the 3D Ken Burns effect [87].

User Control over the 3D Scene. NVS methods attempt to efficiently represent and render a scene while respecting the constraints specified by the input images and their camera parameters. In standard NVS, once the input images are captured, the user has no control over the 3D scene. Among the few exceptions to this custom are Plenoptic Image Editing [111] and Plenopatch [143]. In Plenoptic Image Editing, the user performs editing on one of the input images and the changes will be reflected in all input views and the underlying 3D representation. Plenopatch [143] extends several single-view image editing operations to light fields. Consistently editing a light field is equivalent to editing the underlying 3D scene, as the edits will be reflected downstream in novel views rendered from the edited light field. Providing users control over the 3D scene representation offers more creative freedom and hence improves the utility of NVS in creative digital artistry.

Optimization. Learning-based NVS approaches optimize parameters of a neural network by minimizing photometric and geometric losses, together with regularization terms such as total variation and sparsity. Methods such as MPI [149, 36] perform feed-forward amortized inference of the 3D scene from input images. These methods can be trained once on a large dataset of multi-view images and can generalize to new scenes. On the other end of the spectrum, implicit MLP-based approaches such as NeRF [83] require thousands

(or even more) optimization steps in order to generate a 3D scene representation. Implicit MLP-based techniques, however, do not require large multi-view datasets for training. The time required to optimize implicit scene representations has drastically decreased due to the introduction of efficient hash-based techniques [85]. Nevertheless, these methods still require per-scene optimization of neural models.

1.3 Thesis Overview

Despite the significant advancements in IBR over the past few decades, there are still a number of limitations in existing works. This thesis explores avenues for addressing those limitations by enhancing the *accuracy*, *rendering speed*, and *usability* of IBR systems. Rendering speed and accuracy of existing IBR methods are improved using novel techniques inspired by intuitions about the geometry of the 3D world, powerful deep learning architectures and ideas from IBR literature. Usability is improved by leveraging generative models to reduce the complexity of inputs required to render novel views, and by introducing novel ways with which users can control the 3D scene that is being rendered. These research directions and the key results are summarised below.

Fast and Accurate Novel View Synthesis from Multi-view Stereo

The state-of-the-art IBR method, DeepStereo [36], accurately renders novel views using convolutional networks and plane-sweeping. However, DeepStereo takes around 10 minutes to render a single 500×500 pixel image. Chapter 3 of this thesis introduces a fast and accurate Novel View Synthesis technique, named *FastNVS*. The proposed method takes multi-view stereo imagery as input and produces highly accurate novel views. Similar to the View Interpolation work of Chen and Williams [21], FastNVS decomposes the problem into *proxy scene geometry* estimation and *texture inpainting* (*hole-filling*) tasks. However, unlike the work of Chen and Williams [21], FastNVS uses depth maps instead of pixel correspondences. Moreover, both tasks are cast as self-supervised learning problems that can be solved by training convolutional neural networks over multi-view stereo datasets. FastNVS achieves significant performance improvements over DeepStereo [36] in terms of both speed and accuracy.

Efficient Scene Representation for Spherical Novel View Synthesis

As demonstrated in several prior works [136, 3, 76], modelling appearance with high-dimensional features can help mitigate rendering artifacts, resulting in better view synthesis. However, existing Spherical Novel View Synthesis (SNVS) works [4] rely on Multisphere Images (MSI) representations, which enable fast rendering times but are limited to modelling low-dimensional color values. The high pixel resolution in spherical images coupled with a large number of virtual MSI spheres required for 3D scene representation, makes

using high-dimensional appearance features prohibitively expensive. Chapter 4 introduces a novel MSI representation called Soft Occlusion Multisphere Images (SOMSI) that enables modelling high-dimensional appearance features in MSIs while retaining the fast rendering times of a standard MSI. SOMSI produces considerably better results compared to MSI-based approaches like MODS [4], while having comparably fast rendering times. SOMSI view synthesis quality is on par with state-of-the-art NeRF [83] while being 2 orders of magnitude faster.

Novel View Synthesis from Minimal Scene Specifications

Including FastNVS and SOMSI, all existing IBR approaches require a large number of manually captured input images and pre-processing steps such as performing camera calibration using Structure-from-Motion (SfM). Performing SfM steps requires some level of technical expertise and therefore this requirement excludes the less technical users. Moreover, the requirement of taking several input images implies the scene already exists, therefore, the user cannot freely imagine a novel scene. The need for these requirements is alleviated by introducing two novel techniques that take minimal user inputs: *GVSNet* and *Text2MPI* and allow users to imagine novel 3D scenes.

The GVSNet method proposed in chapter 5 takes a single semantic map as input and renders novel views of the scene in color. GVSNet optionally takes a style image that dictates the global appearance of the scene. The main aim in developing GVSNet is to allow users to imagine and create novel scenes by painting them on a 2D canvas, in the form of a semantic segmentation. As shown in Figure 1.2, GVSNet allows users to create and render scenes from 2D semantic maps. However, manually creating 2D semantic maps can easily become a tedious task. Therefore, Chapter 6 pushes the envelope further by introducing the *Text2MPI* model. Text2MPI maps textual descriptions of a complex real-world scene into an MPI representation. The generated MPIs can then be rendered from novel views; please refer to Figure 1.3 for sample results. Moreover, light fields extracted using the learned MPI models are shown to be realistic enough for training robust implicit scene representation models such as NeRF [83] and InstantNGP [85].
Rendering novel views of a scene from minimal scene specifications such as 2D semantic maps and text requires solving ill-posed problems such as semantics-to-image and text-to-image translation. These problems are further confounded by the need to generate multiple 3D-consistent novel views of the desired scene. These fundamental challenges are overcome by leveraging strong 2D and 3D priors learned by deep generative models. GVSNet and Text2MPI approaches utilize Generative Adversarial Networks [42] and Denoising Diffusion [53, 117] models, respectively.

Granting Users Control over the 3D Scene

IBR approaches provide users almost no control over the 3D scene that is being rendered. For an IBR method, the desired 3D scene is rigidly specified by the input images and their associated camera parameters, as shown in Figure 1.1. The user can move through a 3D

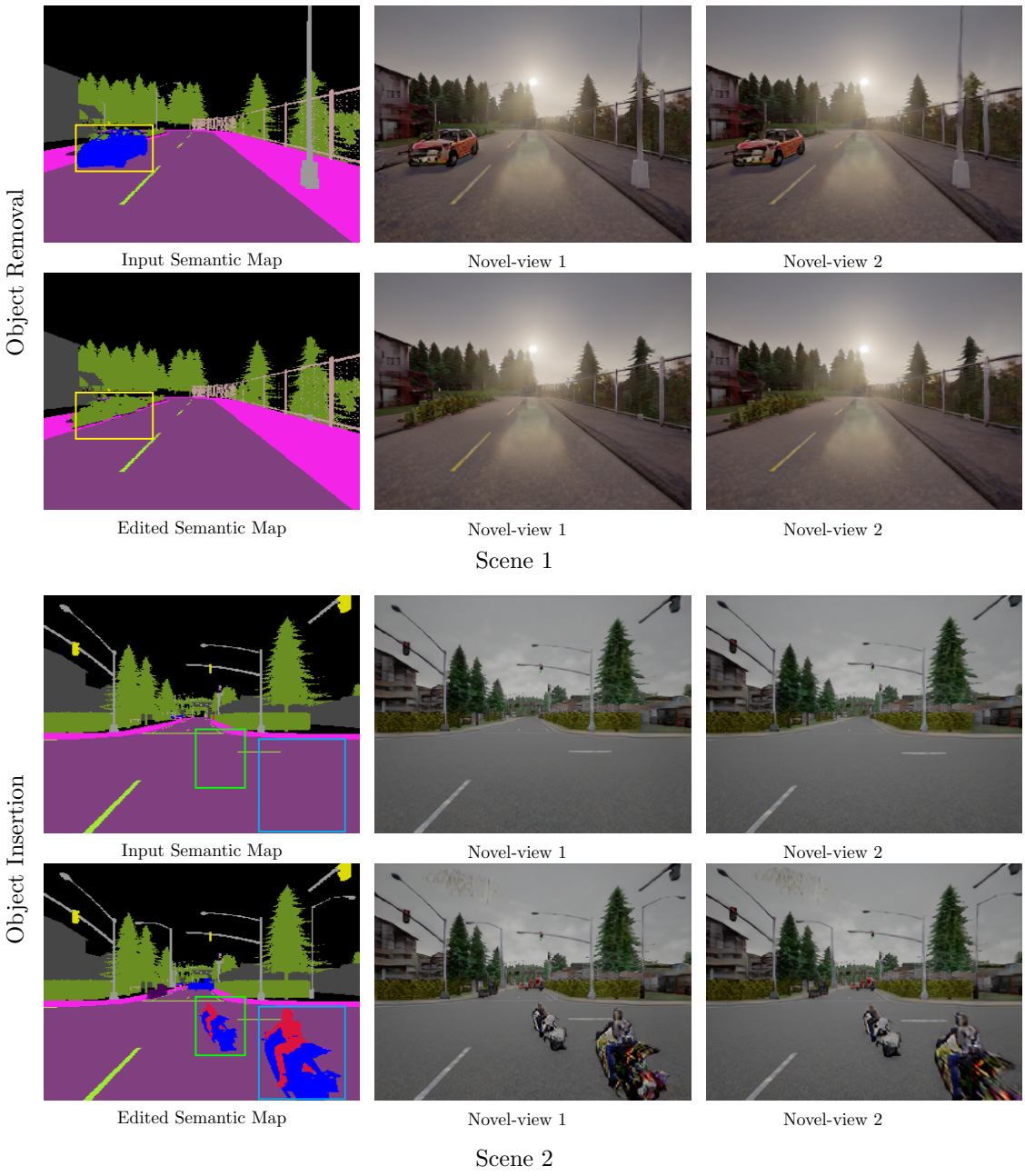


Figure 1.2: **Scene Generation, Rendering and Editing using Semantic Maps.** The semantic maps shown in the first column are used as inputs to render novel views 1 and 2 shown in columns 2 and 3. Scene editing is performed on the input semantic maps: row 1-2 object removal, and in rows 3-4 object insertion. The changes in the input 2D semantic map are reflected in both the underlying 3D scene representation and the resulting novel views.

scene specified by the input views, however, the user cannot alter the scene's geometry and/or appearance. Suppose users want to remove an object from the scene or insert novel objects into the scene, they have to perform a difficult task of editing all input views in a consistent manner. This limitation reduces the creative freedom of IBR users. In contrast, the GVSNet approach grants users the capability to manipulate the 3D scene by simply editing the input 2D semantic map, as shown in Figure 1.2. Similarly, Text2MPI also allows users to change different attributes of the scene by modifying the input text prompts, as shown in Figure 1.3.

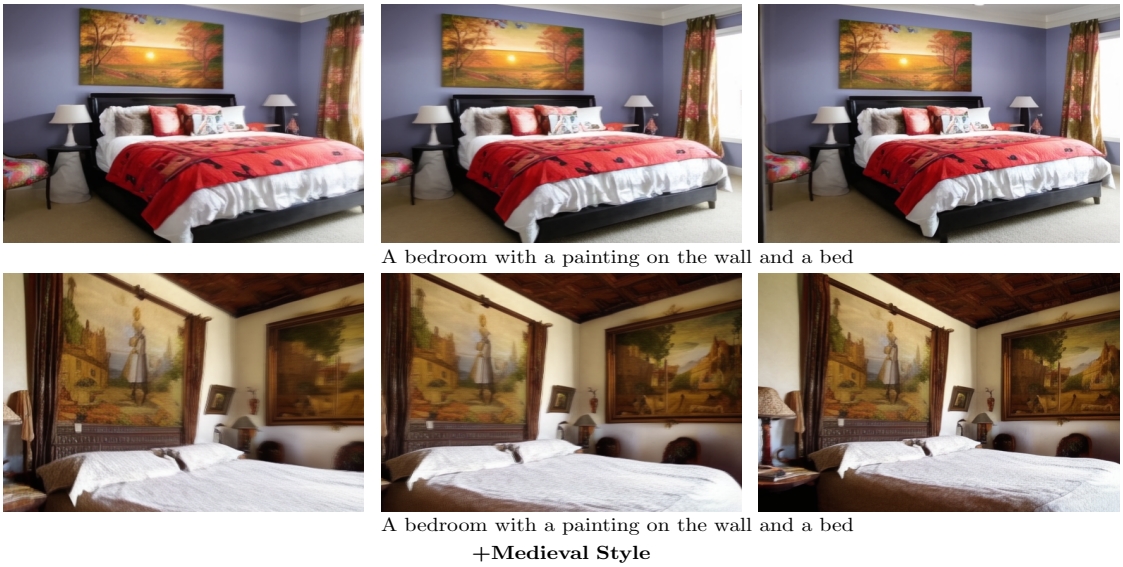


Figure 1.3: **Novel View Synthesis from Text.** Sample novel views generated using Text2MPI. The input text prompts are shown below each scene. As shown in the second row, adding *Medieval Style* to the previous text prompt gives the room a medieval feel.

In summary, this thesis presents four NVS techniques that offer improvements over state-of-the-art methods in terms of speed, accuracy, input complexity and user control. FastNVS significantly improves the speed and accuracy of state-of-the-art NVS. SOMSI presents an affordable method for immersive 360° view synthesis from consumer-grade cameras. SOMSI achieves rendering quality comparable to NeRF [83] while maintaining the rendering speed of MSI-based approaches. GVSNet introduces an NVS technique that takes a single 2D semantic map as input and generates novel view images. Finally, the thesis introduces the Text2MPI model, which converts textual descriptions into MPIs that can be rendered from novel camera poses. GVSNet and Text2MPI require minimal user input and grant users control over the 3D scene.

1.4 Thesis Organization and Contributions

- **Chapter 2** briefly covers the technical background necessary to understand the rest of the thesis. The IBR techniques covered in this chapter are Multiplane Images, Multisphere Images, and Neural Radiance Fields. In addition, generative models such as Generative Adversarial Networks (GANs), Spatially Adaptive De-Normalisation (SPADE), Denoising Diffusion Probabilistic Models (DDPMs), and Latent Diffusion Models (LDMs) are presented.

- **Chapter 3** presents FastNVS, an IBR technique that is significantly faster and more accurate than the state-of-the-art [36], in large-baseline view synthesis. FastNVS decomposes the view synthesis problem into proxy geometry estimation and texture inpainting, which are structured prediction tasks that can be effectively learned via convolutional networks.

- **Chapter 4** presents SOMSI, a spherical view synthesis technique that employs a novel Soft Occlusion Multisphere Images (SOMSI) scene representation. The proposed SOMSI representation effectively scales to encode high-dimensional scene appearance features within the MSI framework by incorporating learnable occlusion layers. SOMSI effectively combines the advantages of traditional MSI-based approaches, such as Matryoshka [4], and MLP-based implicit scene representation techniques, such as Neural Radiance Fields (NeRF) [83]. This combination allows for efficient rendering while maintaining high-quality results.

- **Chapter 5** introduces the Generative View Synthesis (GVS) problem and the proposed GVSNet approach. GVS is a view synthesis task that takes as input a minimal scene description in the form of a 2D semantic map. The proposed GVSNet method provides highly accurate rendering compared to several advanced baselines constructed using state-of-the-art semantics-to-image translation and single-view NVS approaches. Moreover, GVSNet enables users to manipulate the 3D scene geometry by editing the input 2D semantic map and the global appearance features of the scene using a style image.

- **Chapter 6** is dedicated to a model that generates renderable scene representations from text. This chapter introduces Text2MPI, a novel technique that maps textual scene descriptions to MPI representations. Text2MPI makes training feasible by compressing MPIs to a significantly smaller latent representation. Furthermore, Text2MPI leverages the vast generalization abilities of 2D text-to-image models by integrating scene priors generated from pre-trained 2D diffusion models.

- **Chapter 7** concludes the thesis and presents future research directions.

1.5 Publications

A full list of publications is given below. Publications that have directly contributed to this thesis are underlined.

1. Tewodros Amberbir **Habtegebrial**, Stefan Popov, Varun Jampani, Didier Stricker and Vittorio Ferrari
   Text2MPI: Learning to Generate Multiplane Images from Text. Under Review.

2. Tewodros Amberbir **Habtegebrial**, Christiano Gava, Marcel Rogge, Didier Stricker and Varun Jampani
   SOMSI: Spherical novel view synthesis with soft occlusion multisphere images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

3. Tewodros Amberbir **Habtegebrial**, Varun Jampani, Orazio Gallo and Didier Stricker
   Generative view synthesis: From single-view semantics to novel view images. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

4. Tewodros Amberbir **Habtegebrial**, Christian Bailer, Kiran Varanasi and Didier Stricker
   Fast View Synthesis with Deep Stereo Vision. In *VISIGRAPP (VISAPP)*, 2019.

5. Christiano Gava, Mukunda Vishal, Tewodros Amberbir **Habtegebrial**, Federico Raue, Sebastian Palacio, Andreas Dengel
   SphereGlue: Learning Keypoint Matching on High Resolution Spherical Images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2023.

6. Mohamed Selim, Stephan Kraus, Tewodros Amberbir **Habtegebrial**, Alain Pagani, Didier Stricker
   Deep Orientation-Guided Gender Recognition from Face Images. In *12th International Conference on Pattern Recognition Systems (ICPRS)*, 2022.

7. Birhanu Belay, Tewodros Amberbir **Habtegebrial**, Marcus Liwicki, Gebeyehu Belay, Didier Stricker
   A Blended Attention-CTC Network Architecture for Amharic Text-image Recognition. In *International Conference on Pattern Recognition Applications and Methods, (ICPRAM)*, 2021.

8. Birhanu Belay, Tewodros Amberbir **Habtegebrial**, Gebeyehu Belay, Didier Stricker
   Using Automatic Features for Text-image Classification in Amharic Documents. In *International Conference on Pattern Recognition Applications and Methods, (ICPRAM)*, 2020.

9. Birhanu Belay, Tewodros Amberbir **Habtegebrial**, Million Meshesha, Marcus Liwicki, Gebeyehu Belay, Didier Stricker
   Amharic OCR: an end-to-end learning. In *MDPI Applied Sciences*, 2020.

10. Birhanu Belay, Tewodros Amberbir **Habtegebrial**, Gebeyehu Belay, Million Meshesha, Marcus Liwicki, Didier Stricker
    Learning by injection: Attention embedded recurrent neural network for Amharic text-image recognition. In *Preprints*, 2020.

11. Tewodros Amberbir **Habtegebrial**, Gerd Reis and Didier Stricker
    Deep Convolutional Networks For Snapshot Hyperspectral Demosaicking. In *2019 10th Workshop on Hyperspectral Imaging and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, 2019.

12. Birhanu Belay, Tewodros Amberbir **Habtegebrial**, Marcus Liwicki, Gebeyehu Belay, Didier Stricker
    Amharic text image recognition: Database, algorithm, and analysis. In *International conference on document analysis and recognition (ICDAR)*, 2019.

13. Birhanu Belay, Tewodros Amberbir **Habtegebrial**, Marcus Liwicki, Gebeyehu Belay, Didier Stricker
    Factored convolutional neural network for Amharic character image recognition. In *IEEE International Conference on Image Processing (ICIP)*, 2019.

14. Birhanu Hailu Belay, Tewodros Amberbir **Habtegebrial**, Didier Stricker
    Amharic character image recognition. In *IEEE International Conference on Communication Technology (ICCT)*, 2018.

15. Christian Bailer, Tewodros Amberbir **Habtegebrial**, Kiran Varanasi and Didier Stricker
    Fast Feature Extraction with CNNs with Pooling Layers. In *British Machine Vision Conference (BMVC)*, 2018.

2 Preliminaries

In this chapter, we present prior works from the view synthesis and machine learning literature that are crucial in understanding the rest of the thesis. Section 2.1 is dedicated to deep generative models such as Generative Adversarial Networks [42] and Diffusion Models [117, 53], and their applications in conditional image synthesis [88, 101]. Deep generative models are versatile tools that can work with a number of modalities such as text, audio, and images. In this chapter, however, we discuss generative models through the lens of computer vision applications, mainly conditional image synthesis. Finally, in section 2.2, we discuss 3 neural rendering techniques that are closely related to this thesis: Multiplane Images (MPI) [149], Multisphere Images (MSI) [4] and Neural Radiance Fields (NeRF) [83].

Contents

2.1 Generative Models **24**

2.2 Neural Rendering **31**

2.1 Generative Models

Generative Models are a class of probabilistic models aimed at learning the probabilistic distributions behind real-world data such as images, audio and text. In most cases, these real-world distributions are highly complex and are often approximated with learned distributions parameterized via deep neural networks. Given a finite set of observations $\{\mathbf{x}_i\}_{i=1}^{m}$, generative models aim to learn the underlying distribution $p(x)$ that generates the observations. Generative models are useful in downstream tasks to generate new samples from a learned distribution, and perform likelihood estimation on existing samples. In computer vision, conditional generative models of the form $p(x|c)$, are also common. Some examples are modeling a distribution over images conditioned on class labels, which can be used to generate class-conditional samples, *e.g.* $x_{cat} \sim p(x|c = cat)$ or $x_{dog} \sim p(x|c = dog)$. Other examples include generating images conditioned on semantic segmentation and textual descriptions, and generating monocular depth conditioned on images.

In this section we will present a brief review of deep generative models such as Generative Adversarial Networks (GANs) and Diffusion Models (DM), and their applications in image synthesis [88, 62, 133, 101].

2.1.1 Generative Adversarial Networks

Generative Adversarial Networks [42] are generative models that learn a data distribution by training two models that play a min-max game against each other. The first model is a generator, G, that tries to approximate the distribution $p(x)$. The second model is a discriminator, D, trained with an objective to correctly distinguish real samples from those generated by the generator. In GANs, the generator is modeled as a mapping from a prior distribution $p(z) \sim N(0, I)$ to data samples $\mathbf{x} \sim p(x)$. The noise samples are d dimensional vectors sampled from a standard Gaussian with zero mean and unit variance.

The generator and discriminator models are trained using the following objective:

$$\min_{\theta_G} \max_{\theta_D} \mathbb{E}_{\mathbf{z}\sim p(z)} \left[\log(1 - D(G(\mathbf{z})))\right] + \mathbb{E}_{\mathbf{x}\sim p_{data}} \left[\log(D(\mathbf{x})\right] \tag{2.1}$$

where θ_D and θ_G are parameters of the networks D and G, respectively.

During training, the generator model aims to minimize the min-max objective shown in Equation 2.1, while the discriminator tries to maximise that same objective. During training these models are trained in turns, as shown in Algorithm 1. At each training step, the generator tries to create samples $G(\mathbf{z})$ that are similar to the real samples $\mathbf{x}_{real}$. The generator minimizes the loss $\mathcal{L}_G$ which pushes the discriminator output $D(G(\mathbf{z}))$ towards 1. On the other hand, the discriminator tries to differentiate generated samples from real samples by pushing $D(G(\mathbf{z}))$ towards 0 and $D(\mathbf{x}_{real})$ to 1. In practice, the generator is updated once for every k discriminator steps; k is often in the range 1 to 5. As demonstrated in Goodfellow *et al.* [42], given sufficiently complex models D and G, the

Algorithm 1 Training Generative Adversarial Networks

while $i < N$ **do**
  $\mathbf{x}_{real} \sim p(\mathbf{x}), \quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ ▷ Get a real sample $\mathbf{x}_{real}$ and noise $\mathbf{z}$
  $\mathbf{x}_{fake} = G(\mathbf{z})$
  if $(i \mod k) == 0$ **then**
    $\mathcal{L}_G = \log(1 - D(\mathbf{x}_{fake}))$
    $\theta_G \leftarrow \theta_G - \gamma \nabla \mathcal{L}_G(\theta_G)$ ▷ Gradient Descent
  else
    $\mathcal{L}_D = \log(1 - D(\mathbf{x}_{fake})) + \log(D(\mathbf{x}_{real}))$
    $\theta_D \leftarrow \theta_D + \gamma \nabla \mathcal{L}_D(\theta_D)$ ▷ Gradient Ascent
  end if
  $i = i + 1$
end while

min-max game in Algorithm 1 can reach a state of equilibrium. At equilibrium, the output of D approaches $\frac{1}{2}$ for both real and generated samples thus implying $G(\mathbf{z}) \approx p(\mathbf{x})$.

Application: Semantic Image Synthesis

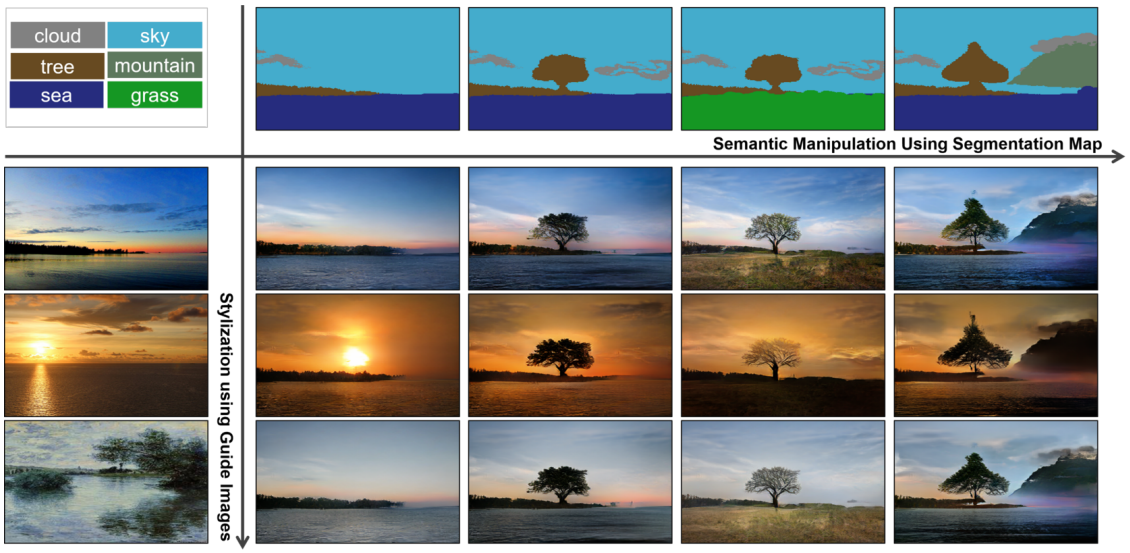


Figure 2.1: **The Semantic Image Synthesis (SIS) Problem.** Example SIS results, the semantic maps in the top row are rendered in color using different style images shown in the first column. Each image has the content described by the semantic map and follows the global appearance characteristics of the style image. Image source: Park *et al*. [88]

Semantic Image Synthesis (SIS) is a problem where a photorealistic image is generated

from an input semantic segmentation and a style guidance image, see Figure 2.1. Semantic Image Synthesis falls under the umbrella of image-to-image translation (I2I) problems, where an image is translated into another pixel-aligned image. The input and output are images following different styles or they can also be in different modalities. Examples of I2I problems include style transfer, image colorization, thermal to color image conversion. In this section, we present a brief overview of one of the most popular SIS techniques called SPADE [88].

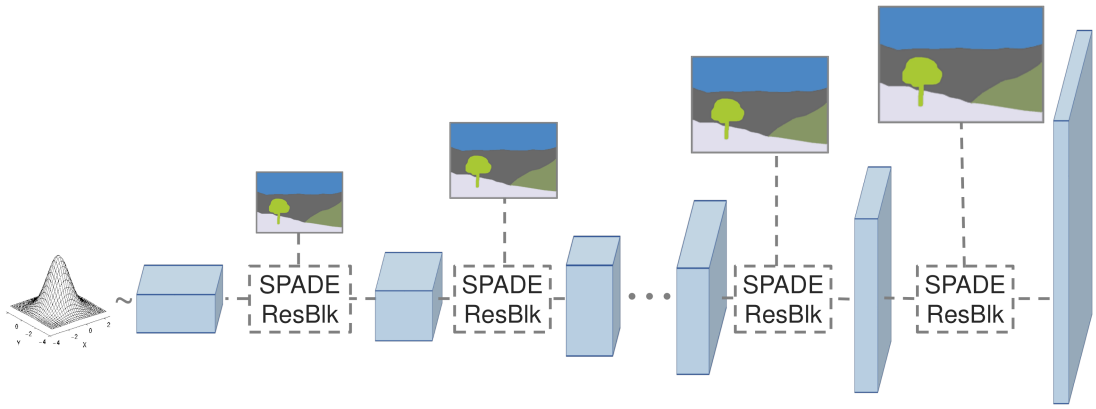


Figure 2.2: **SPADE Generator Architecture.** The SPADE generator is composed of multi-scale SPADE ResBlock. Each block takes an input semantic segmentation map and convolutional activations from a preceding block upsampled by a factor of 2. The semantic map is used to modulate the convolutional features via Spatially Adaptive DeNormalisation layers 2.3. A VAE latent embedding of style image is also given as input, to the block operating at the lowest spatial resolution. Image source: Park *et al*. [88]

As shown in Figure 2.2, the SPADE [88] semantic image synthesis architecture is multi-scale convolutional architecture that generates a color image from a semantic map and a latent embedding representing a style image. The style image is represented via a latent embedding vector sampled from Variational Autoencoder (VAE) model [70]. During training the method requires a large dataset of semantic segmentation maps and their corresponding color images. In the training phase, the color image serves as a style image. However, after training the model takes a semantic map and an arbitrary style image as demonstrated in Figure 2.1.

SPADE is trained as a GAN, consisting of a generator model shown in Figure 2.2 and a UNet style discriminator network (which we omit for brevity). The SPADE generator architecture is composed of *SPADE Res-Blocks* which contains a SPADE layer followed by a rectified linear unit and convolution layers, see Figure 2.3. SPADE layers apply normalization using scale and shift parameters computed from the input semantic map. Prior SIS works use normalization layers whose parameters are not spatially adaptive *i.e*. the same (de)normalization parameter is shared across all pixels in a feature map. As

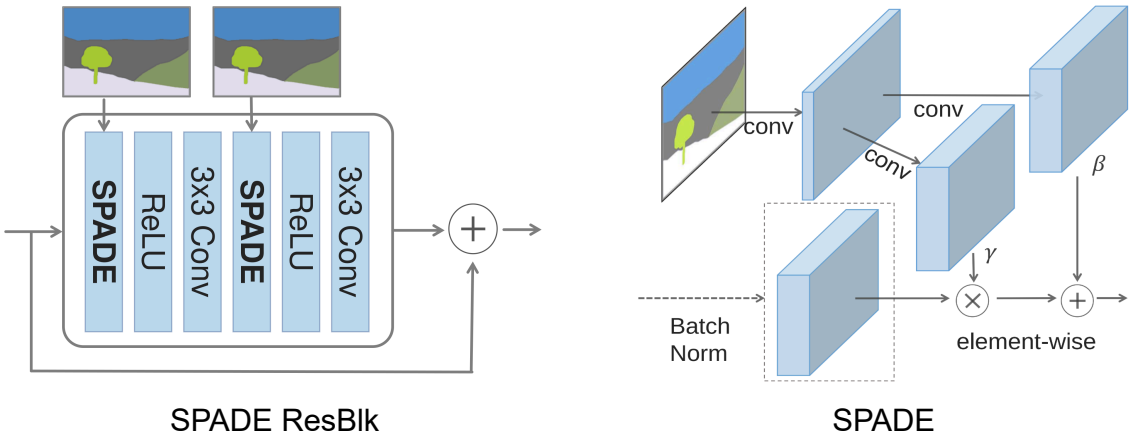


Figure 2.3: **SPADE Res-Blocks and the SPADE Layer.** A SPADE ResBlock (left) processes incoming convolutional features with convolution, ReLU and SPADE layers. Before each convolution the feature maps are modulated through a SPADE layer (right). The SPADE layer applies pixel-wise normalization using scale (γ) and shift (β) parameters extracted from semantic features. The effect of spatially adaptive normalization with a SPADE layer on the generated images, is shown in Figure 2.4. Image source: Park *et al*. [88]

shown in Figure 2.4, by using the semantic information during normalization, SPADE avoids problems exhibited by previous works. Prior works such as Pix2Pix [133] and Pix2PixHD [133], generate uniform outputs that lack realistic texture. Unlike the simplified GAN model discussed earlier in this section, the SPADE model is trained using GAN losses and additional photometric losses computed over the VGG [115] features of a predicted image and its corresponding ground-truth image.

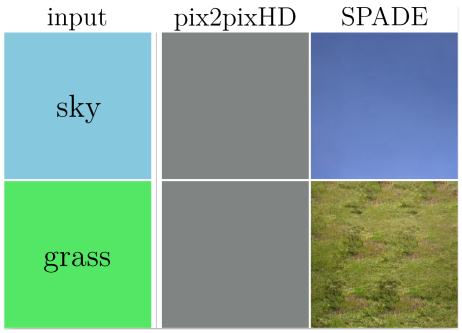


Figure 2.4: **Loss of Semantic Information in SIS Networks.** The SPADE layer proposed in [88] creates spatially varying texture while previous works struggle to generate plausible textures. Image source: Park *et al*. [88]

2.1.2 Diffusion Models

Diffusion Models (DMs) [53, 117, 101] are a special family of generative models, with a hierarchical encoder-decoder structure. The encoder is a stochastic process with fixed parameters that progressively injects noise to samples. More precisely, the encoder maps an observation $\mathbf{x}_0$ into latent variables $z_1, z_2, \ldots, z_T$, as

$$z_t = \sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon_0 \tag{2.2}$$

where $\epsilon_0 \sim \mathcal{N}(0, I)$, $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$, $\alpha_t = 1 - \beta_t, \sigma_t^2 = \beta_t$.

The signal-to-noise ratio at each time step t is controlled with a set of pre-defined noise schedule parameters, $\{\beta_t\}_{t=1}^{T}$, where $\beta_{i+1} < \beta_i \quad \forall i, j \in [1, 2, \ldots, T]$. Values of βs are chosen such that $\mathbf{z}_0 = \mathbf{x}_0$ and the last sample $\mathbf{z}_T$ approaches a sample from a standard normal distribution. For a visual demonstration of the encoder/forward and decoder/backward processes see Figure 2.5.

The decoder is a multi-step denoising process that uses learned distributions $p(\mathbf{z}_{t-1}|\mathbf{z}_t)$. In practice a time conditioned neural network, $\mathcal{P}(\mathbf{z}_t, t)$, is used to approximate parameters of distributions $p(\mathbf{z}_{t-1}|\mathbf{z}_t)$.

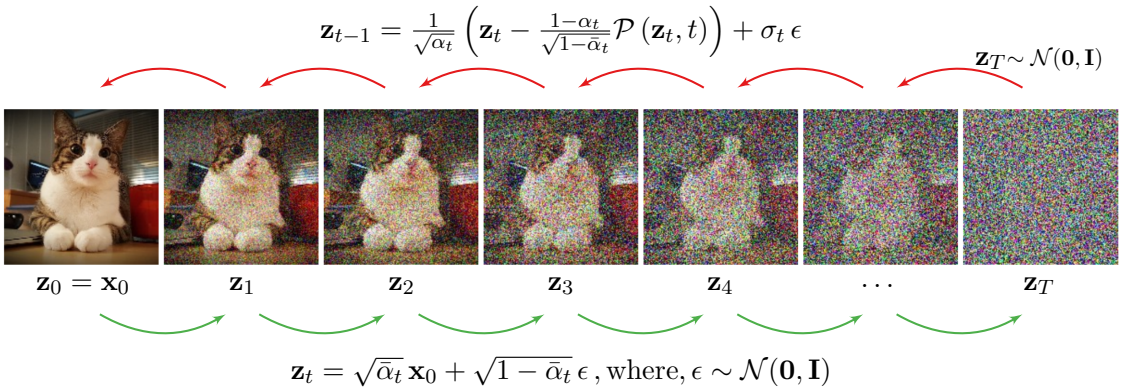


Figure 2.5: **Forward and Backward Diffusion Processes.** The forward/encoder process is a time dependent stochastic process that adds small amount of noise to latent variables in such a way that the last latent variable $\mathbf{z}_T$ reaches a standard Gaussian distribution. The backward/decoder process performs the opposite task *i.e.* latent variables are progressively denoised so that the latent variable at $t = 0$ resembles a sample from the real data distribution. The backward process is also a Gaussian process whose mean is determined via a neural network and uses the same variance schedule as the forward process. Photo credits: Arash Vahdat

As demonstrated in [53] a diffusion model parameterized via a neural network, $\mathcal{P}$, can

be trained by minimizing the following objective function:

$$\mathop{\mathbb{E}}_{\mathbf{x}_0 \sim p(x),\ t \sim \mathcal{U}(0,T)} ||\epsilon_0 - \mathcal{P}(\sqrt{\bar{\alpha}_i}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon_0)||^2 \tag{2.3}$$

As shown in Algorithm 2, during training the objective shown in Equation 2.3 is minimized to train a neural network. The neural network gets a noisy image $\mathbf{z}_t = \sqrt{\bar{\alpha}_i}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon_0$ and it estimates its noise component ϵ_0. There is an equivalent formulation of diffusion models where the neural network estimates the signal component of $\mathbf{z}_t$ instead of noise. The two formulations are mathematically equivalent. However, in practice, ϵ prediction networks tend to produce more realistic images, and are more commonly used.

During inference, shown in Algorithm 3, a diffusion model maps a noise sample $\mathbf{z}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, into a clean sample, $\mathbf{z}_0$, by sampling from the distribution $p(\mathbf{z_{t-1}}|\mathbf{z_t})$ for time steps $T, T-1, \ldots, 1$, as shown in Equation 2.4.

$$\mathbf{z}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{z}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\mathcal{P}\left(\mathbf{z}_t, t\right)\right) + \sigma_t\,\epsilon$$
$$\text{where } \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{2.4}$$

The training and inference steps are summarized below in Algorithms 2 and 3.

Algorithm 2 Diffusion Models: Training

while $i <$ max_iterations **do**
  $\mathbf{x}_0 \sim p(\mathbf{x})$ ▷ Sample an image from the dataset
  $\epsilon_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $t \sim \mathcal{U}(0, T)$
  $\mathcal{L} = ||\epsilon_0 - \mathcal{P}(\sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon_0)||^2$
  $\theta_{\mathcal{P}} \leftarrow \theta_{\mathcal{P}} - \gamma\,\nabla\,\mathcal{L}(\theta_{\mathcal{P}})$ ▷ SGD on parameters of $\mathcal{P}$, with learning rate γ
  $i = i + 1$
end while

Algorithm 3 Diffusion Models: Sampling

$\mathbf{z}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for t in $[T, T-1, \ldots, 1]$ **do**
  $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$ else $\epsilon = \mathbf{0}$
  $\mathbf{z}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{z}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\mathcal{P}\left(\mathbf{z}_t, t\right)\right) + \sigma_t\,\epsilon$
end for
return $\mathbf{z}_0$

Application: High Resolution Text to Image Translation

Latent Diffusion Models (LDMs) [101] are a special type of diffusion models that are trained on latent image representations. LDM learn to generate high resolution photorealistic images from textual descriptions. Unlike recent image generation models that work directly in the image space, LDMs operate on a compressed and more efficient space of latent images. Latent images are generated from an auto-encoder network that is trained to map input images into lower dimensional latent representations and reconstruct them into full resolution images.

LDMs were first proposed by Rombach *et al.* [101] as a conditional generative model for text conditioned image generation. As shown in Figure 2.6 the forward diffusion process uses a latent representation. Given an image $\mathbf{x} \in \mathbb{R}^{H \times W \times 3}$, the latent encoder model $\mathcal{E}$ maps it into its corresponding latent representation with dimensions $\frac{H}{k} \times \frac{W}{k} \times 4$. In practice $k = 8$ resulting in ≈ 64 times smaller memory and compute cost. Similarly, during the inference/sampling phase a Gaussian noise with the same dimensions as the latent images is sampled and progressively denoised into a clean latent image representation using the sampling Algorithm 3. The final color image output is computed by decoding the denoised latent representation into color using the pretrained latent image decoder $\mathcal{D}$, shown in Figure 2.6. In both the training and inference phases, the input text is used to condition the denoising process using cross attention layers that combine the convolutional image features with the text features.

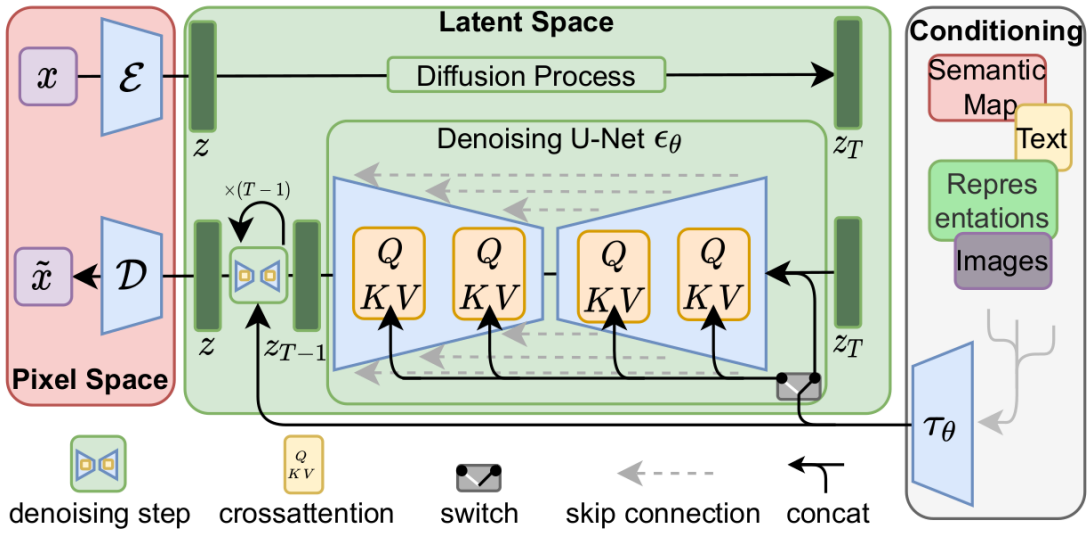


Figure 2.6: **LDM Network Architecture.** Image source: Rombach *et al.* [101]

2.2 Neural Rendering

Deep neural networks have become an indispensable tool in learning-based view synthesis works. Neural Rendering is a common term that refers to a wide variety of image synthesis algorithms that use neural networks in their pipeline. In this section we will review two of the most popular neural rendering methods: Multiplane Images [149, 128] and Neural Radiance Fields [83].

2.2.1 Multiplane Images

Representation. A Multiplane Image (MPI) [149] approximates a 3D scene with a set of textured $RGB\alpha$ planes arranged at different depths. An MPI $\mathcal{M}$ is a 4D tensor of size $D \times H \times W \times 4$, where D is the number of planes, H and W are the image height and width, and the final dimension represents color and alpha channels. Each plane in the MPI has an RGB texture map and an alpha channel that indicates the opacity of the plane. The planes are arranged in ascending depth order, with the closest plane at z_{near} and the farthest at z_{far}. As shown in Figure 2.7, an MPI can be visualized as a stack of transparent planes in 3D. In multi-view MPI methods, the scene bounds z_{near} and z_{far} are determined automatically from Structure-from-Motion (SfM) information [149], using the closest and farthest detected 3D points in the reference image. However, in case when the MPI is built from a single image [128], these values are typically chosen by the user or determined heuristically using geometric assumptions. In large-scale scenes with significant perspective effects, selecting plane depths by linearly sampling the inverse depth range counteracts the effect of perspective projection, where object displacement is inversely proportional to depth. However, for scenes with limited depth ranges, linearly sampling the z_{near} to z_{far} range is sufficient.

Rendering. Given an MPI representation, $\mathcal{M}$, we are interested in rendering a novel view from an arbitrarily chosen camera pose $[R|\mathbf{t}]$. Suppose the camera pose is defined as an $SE(3)$ transformation matrix, from the reference camera (where the MPI planes are defined) to the target novel view camera pose. Let's denote the reference and novel view camera intrinsics as K_r and K_{nv}. MPI rendering is performed in two steps: *homography warping* the MPI planes from the reference to the target camera pose, and back-to-front *alpha compositing* of the warped MPI plane textures.
Homography Warping: The MPI planes are warped to the target camera using homography matrices computed as follows [46]:

$$H_i = k_r \left(R + \frac{R^T \mathbf{t}\, \mathbf{v}\, R^T}{-z_i + \mathbf{v}\, R^T \mathbf{t}} \right) k_{nv}^{-1} \tag{2.5}$$

where $\mathbf{v} = [0, 0, -1]$ is the plane normal.

Given homographies $\{H_i\}_{i=1}^{D}$ the MPI planes can be warped to the novel view. Warping starts by mapping every target camera pixel, $\mathbf{p}_t = (x_t, y_t)$, to its corresponding sampling

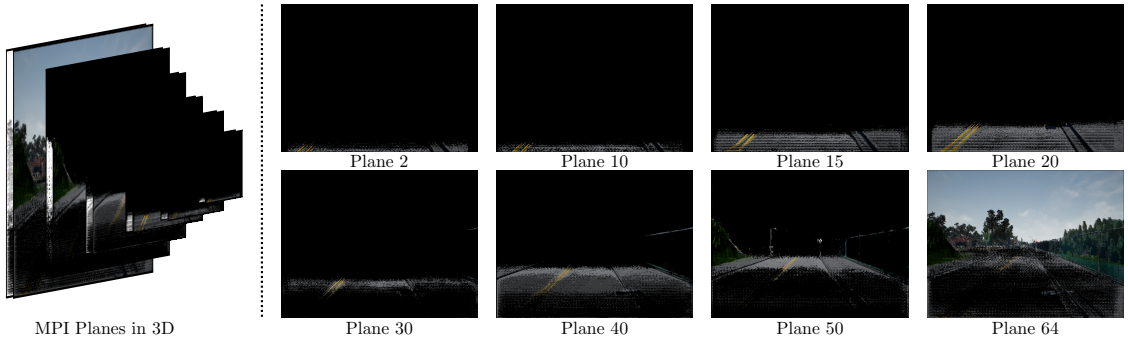


Figure 2.7: **MPI Scene Representation.** Subsets of textured planes taken from an MPI scene representation that uses 64 planes. The left side illustrated MPI planes placed at different depth values in 3D. On the right, we show the MPI color images multiplied by their respective opacity values. As the depth of the plane increases the opaque part of each plane recedes towards faraway objects in the scene.

location (u, v) in the reference camera using Equation 2.6. As sampling locations are floating points, the color values of $\mathbf{p}_t$ are computed by blending the four nearest neighbours of reference sampling location (u, v), using a bilinear sampling kernel. Note that, bilinear sampling is a differentiable operation [63], making it suitable for use in learning-based approaches.

$$\begin{aligned} [\bar{u}, \bar{v}, \bar{w}]^T &= H_i \ \left[x_t \, , y_t \, , 1\right]^T \\ u, \ v &= \bar{u} \ / \ \bar{w} \, , \bar{v} \ / \ \bar{w} \end{aligned} \tag{2.6}$$

Back-to-front Alpha Composition: After applying homography warping operation $\mathcal{W}$ over the reference MPI representation, $\mathcal{M}_{nv} = \mathcal{W}(\mathcal{M}, \{H_i\}_{i=1}^{D})$, the warped MPI planes are back-to-front alpha composited to compute the final novel view image, as shown in Equation 2.7. Let's separate the color and opacities of the warped MPI $\mathcal{M}_{nv}$ into $\hat{c}$ and $\hat{\alpha}$, respectively.

$$c_{nv} = \sum_{i=1}^{d} \left[\hat{c}_i \, \hat{\alpha}_i \prod_{j=1}^{i-1} (1 - \hat{\alpha}_j) \right] \tag{2.7}$$

Learning Multiplane Images We will use the Stereo-Magnification (SM) approach from Zhou *et al*. [149] as our choice of an MPI model. SM is a learning-based view synthesis approach that generates an MPI scene representation of complex real-world scenes given two input views. As shown in Figure 2.8, the input to SM pipeline is a plane sweep volume built by warping the second input view onto the first/reference view via planar homography warpings. The neural network, $\mathbf{f}$, is a convolutional network that takes the

input plane-sweep volumes and predicts an MPI scene representation. The predicted MPI representation is then rendered using homography and alpha compositing operations as discussed above.

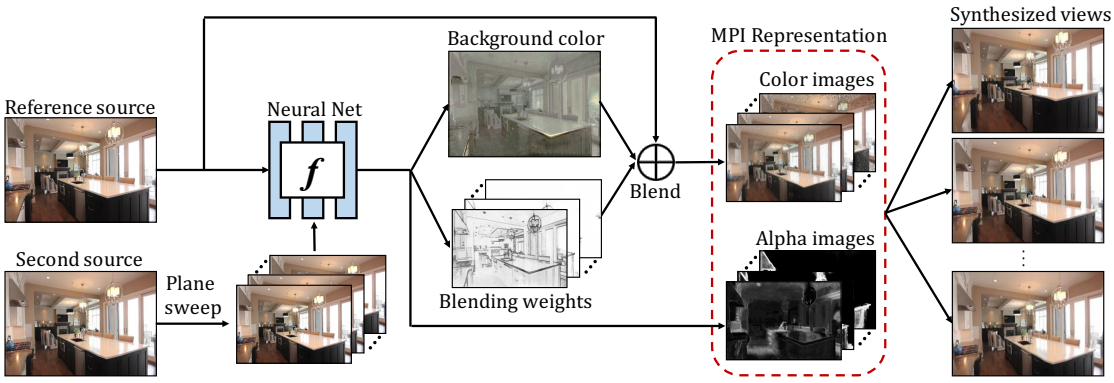


Figure 2.8: **Learning Multiplane Image Representations.** The Stereo Magnification (SM) approach uses a convolutional network (denoted as $\mathbf{f}$), to estimate an MPI from a plane-sweep volume built from two input images. The network predicts per-plane opacities $\{\alpha\}_{i=1}^{D}$ and blending weights $\{\beta\}_{i=1}^{D}$, and a background image b. Per-plane color images are computed by linearly combining the reference image a and the background image via the plane-wise blending weights, as $c_i = a\,(1-\beta_i) + b\,\beta_i$. Note, that this is an architectural design decision; it is possible to estimate MPI color directly with the model $\mathbf{f}$. The plane-wise color and opacity images make up an MPI scene representation, which can be rendered from novel views using homography warping and alpha compositing operations. Image Source: Zhou *et al*. [149]

The stereo magnification model is trained on a dataset of posed multi-view images collected from YouTube video clips. Each training sample consists of 3 views with known intrinsics and extrinsic camera parameters. The first 2 views are used as input for SM model, while the third view is held out to serve as a target novel view. The SM model is trained by minimizing the discrepancy between the held out target novel view image and an estimated novel view image.

2.2.2 Multisphere Images

Multisphere Images (MSI) [11, 4] are introduced as a spherical extension of Multiplane Images (MPI), serving as suitable representations for spherical view synthesis by replacing MPI planes with concentric spheres. Similar to MPI planes, each MSI sphere has an associated color and opacity texture map, represented in the form of an Equirectangular Projection map. In this section, we will review the MSI representation, its rendering algorithm, and the spherical camera model.

Spherical Camera Model. A spherical image is an environment mapping that captures the entire visible scene from a single point in space. In other words, it is a plenoptic function containing the color values of rays passing through the 3D point in any direction. Spherical images are usually stored as a 2D pixel map of dimensions $h \times w$, often referred to as Equirectangular Projection (ERP). Each ERP pixel $\mathbf{p} = [u, v]^T$, where $u \in [0, w-1], v \in [0, h-1]$, corresponds to a point on a unit sphere $[\theta, \phi, 1]^T$ expressed in spherical coordinates. This mapping between a point on the unit sphere and its pixel location on the ERP is given as

$$u = w\,(1 - \frac{\theta}{2\,\pi}), \qquad\qquad v = h\,\frac{\phi}{\pi}, \tag{2.8}$$

with x-axis pointing into the screen, y-axis to the left and z-axis up; a Cartesian point $\mathbf{x} = [x, y, z]^T$ is converted into spherical coordinates $\mathbf{x}^s = [\theta, \phi, r]^T$ as

$$\theta = atan(\frac{y}{x}) \qquad\qquad \phi = acos(\frac{z}{r}) \qquad\qquad r = \sqrt{x^2 + y^2 + z^2}. \tag{2.9}$$

A cartesian point $\mathbf{x}$ in 3D can be projected on to an ERP by first obtaining $\mathbf{x}^s = [\theta, \phi, r]^T$ with Equation 2.9, followed by mapping $\mathbf{x}^s$ to the ERP location $[u, v]^T$ via Equation 2.8.

MSI Representation MSI Rendering.
MSI Representation: An MSI is a set of concentric spheres of radii $\{r_i\}_{i=1}^d$ with each sphere representing a spherical image. The r_i values are set by linearly sampling the inverse depth range between predefined near r_{near} and far r_{far} values. Each of the d spheres in an MSI has transparency $\alpha \in [0, 1]^{m \times 1}$ and color $C \in [0, 255]^{m \times 3}$, where $m = h \times w$ is the pixel resolution of the spherical images and d is the number of spheres. The number of spheres controls the trade-off between the fidelity of the scene representation and the computational costs (rendering time and memory). Increasing d yields higher fidelity but requires more memory and results in slower rendering.
MSI Rendering: Figure 2.9 shows an illustration of MSI spheres centered around reference view $\mathbf{r}$. Suppose we want to render a spherical image from a target camera pose $\mathbf{t}$. For a ray direction $\mathbf{r}$ (corresponding to a location $\mathbf{p} = [u, v]^T$ in the target ERP), we first compute the ray's intersections $\mathbf{x}_i^p \in \mathbb{R}^3$, $\forall i \in \{1, ..., d\}$ with the d MSI spheres using a standard sphere-ray intersection technique [91]. We then record MSI transparency and color values at these intersection points: $\{(\alpha_i^p, c_i^p)\}_{i=1}^d$. The final color C^p for target ray $\mathbf{p}$ is calculated by over-compositing c_i^ps with α_i^ps in a back-to-front manner (also known as alpha compositing):

$$c^p = \sum_{i=1}^{d} \alpha_i^p\, c_i^p \prod_{j<i} (1 - \alpha_j^p). \tag{2.10}$$

The same process is used for all the ray directions in the target view to obtain final novel view spherical image at the target location. Since this process is trivially parallelizable, it is straightforward to use GPUs for fast rendering of novel views. One could also export MSI spheres as textured meshes to leverage standard rendering engines for real-time novel

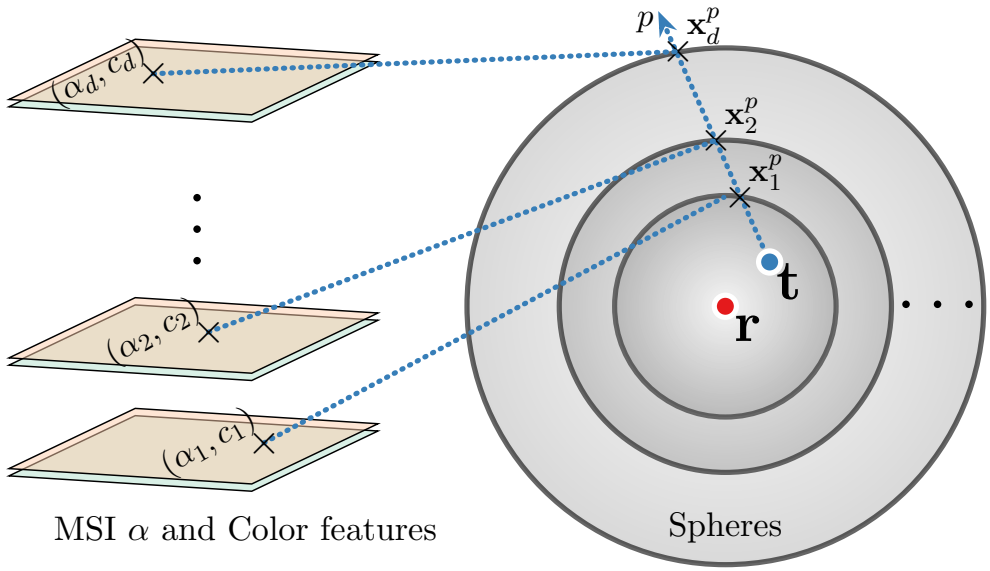


Figure 2.9: **MSI Representation and Rendering.** An MSI is composed of concentric spheres centered at a reference camera position r. Each sphere has an associated color and opacity ERPs as shown here. Color values for pixel $\mathbf{p}$ at a target novel view pose t are computed as follows. First, we shoot a ray through $\mathbf{p}$ into the scene and record the intersections of the ray with each sphere, denoted as $\mathbf{x}_i^p$. The intersection points are projected to the reference ERPs, and are used to bilinearly sample color and opacities from the ERPs. The sampled color and opacity values then back-to-front alpha composted to produce the final color values for pixel $\mathbf{p}$.

view rendering.

Learning MSI Scene Representations The MatryODShka [4] spherical view synthesis approach, illustrated in Figure 2.10, learns an MSI scene representation from a pair of spherical images. Under spherical-world assumptions, the two input views are warped into a sphere-sweep volume. A convolutional neural network is then used to estimate an MSI scene representation from the input sphere-sweep. The MSI is then rendered from novel camera poses using the MSI rendering technique discussed above.

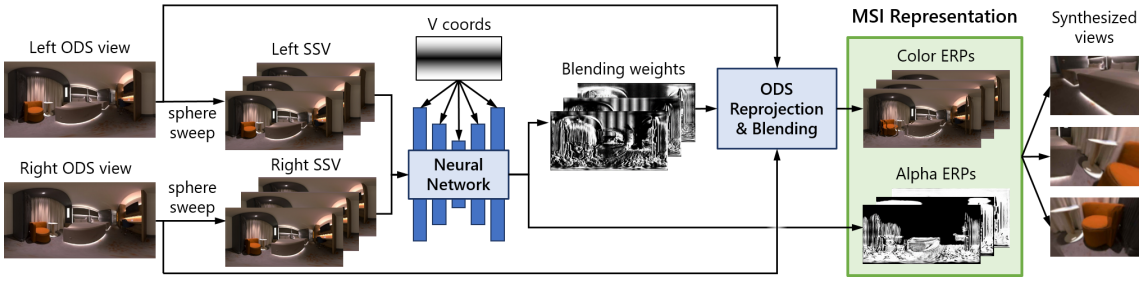


Figure 2.10: **MODS Network Architecture.** Image Source: Attal *et al*. [4]

2.2.3 Neural Radiance Fields

MPI- and MSI-based approaches can estimate discrete scene representations from a few input images. Given large datasets of posed multi-view images, these models can be trained to generalise to a wide range of unseen test scenes. Moreover, the inference and rendering stages on MPIs and MSIs are extremely fast, enabling interactive frame rates on machines with modern GPUs. However, these models have strong limitations due to their scene representation structures and training strategies.

- The discrete nature of their scene representations lead to rendering artifacts due to the Nyquist sampling limit. In MPIs and MSIs, only novel views at camera poses that are close to the reference can be rendered reliably.
- Single MPIs and MSIs cannot be used to perform view synthesis tasks that require complex camera motion patterns, such as flying a camera around an object in the scene. MPIs and MSIs are suitable for simpler scenarios such as light field interpolation, where the camera movement is restricted roughly to a plane or a small volume 3D with limited depth.
- MPI/MSI-based approaches are usually trained with a few fixed number of input images, and increasing the number of input images requires retraining the network.

Mildenhall *et al*. [83] introduced the Neural Radiance Field (NeRF) approach of scene representation and an accompanying volumetric rendering technique. NeRF produces novel views that are significantly more accurate than that of MPI-based view synthesis approaches. NeRF uses a multi-layer perceptron (MLP) to represent a desired scene, enabling a more general scene representation that is not limited by discrete plane like structures used in MPI-based works. In this section we will briefly review the NeRF approach including how NeRF represents a scene with a single MLP, volumetric rendering of NeRF MLPs, and the training strategy.

NeRF Scene Representation NeRF uses a single MLP network, $\mathbf{F}_\Theta$, to encode information about a scene's geometry, texture and light. The MLP is designed as function that takes 5D query vectors (x, y, z, θ, ϕ) and returns the density σ of the 3D point at (x, y, z),

together with color and intensity of light emanating from this 3D point in the direction (θ, ϕ), $(RGB\sigma) \leftarrow \mathbf{F}_\Theta((x, y, z, \theta, \phi))$, see the illustration in Figure 2.11 (a).

NeRF Rendering. NeRF rendering works by building a plenoptic function that produces color values associate with any ray $\mathbf{r}$ in the 3D scene. Color values of a novel view pixel $\mathbf{p} = (u, v)$ are computed by first estimating the ray $\mathbf{r}$ passing through $\mathbf{p}$ into the scene, and computing the color associated to the ray. The color $\mathbf{c}_r$ associated with $\mathbf{r}$, is computed querying the MLP model for color and opacity values of points along the ray, and performing back-to-front compositing on the resulting color and opacities, as shown in Algorithm 4.

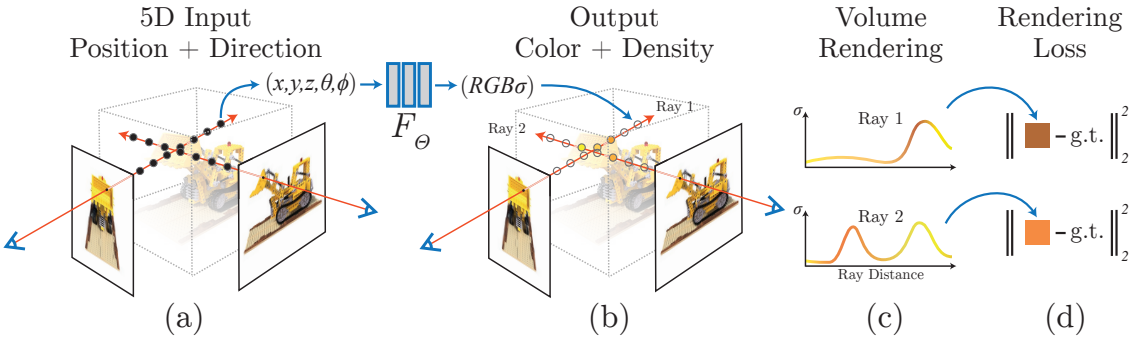


Figure 2.11: **NeRF Scene Representation and Rendering Pipeline.** NeRF represents a scene using an MLP network. Novel view synthesis are volume rendered using color and opacity values computed by querying the scene representation MLP network. Image Source: Mildenhall *et al.* [83]

Let's take a closer look at the process of rendering a novel view at a target camera, whose extrinsic and intrinsic parameters are given as $[R, \mathbf{c}]$ and K, respectively. Let's parameterise rays associated to each pixel in this camera as follows, $\mathbf{r} = \mathbf{o} + t\,\mathbf{d}$, where $\mathbf{o}$ center of the novel view camera center $\mathbf{c}$, and $\mathbf{d}$ is a unit direction vector computed as shown in Equation 2.11.

$$\begin{aligned} \bar{\mathbf{d}} &= K^{-1} \times [u, v, 1]^T \\ \mathbf{d} &= R\,[\bar{\mathbf{d}} \,/\, ||\bar{\mathbf{d}}||^2] \end{aligned} \tag{2.11}$$

Viewing direction parameters (θ, ϕ) are computed applying cartesian to spherical coordinate conversion on the unit vectors $\mathbf{d}$, see Equation 2.9. The N 3D query points $\{\mathbf{x}_i\}_{i=0}^{N-1}$ are computed as $\mathbf{x}_i = \mathbf{o} + t_i \times \mathbf{d}$, using uniformly distributed sampling locations $\{t_i\}_{i=0}^{N-1} = Uniform(t_{near}, t_{far}, N)$. Near and far scene bounds are calculated from 3D points clouds of the scene computed with Structure-from-Motion (SfM) [108].

Positional encoding is applied to the input query values $(x_i, y_i, z_i, \theta, \phi)$, where each input is encoded using sinusoidal functions with K frequency levels, as shown in Equation 2.12.

$$enc(x) = (sin(2^0 \pi\, x)), (cos(2^0 \pi\, x)), \ldots, (sin(2^{K-1} \pi\, x)), (cos(2^{K-1} \pi\, x)) \quad (2.12)$$

Encoding inputs of the NeRF MLP using multi-scale sinusoidal functions is proven to be effective in generating high frequency details, as shown in Figure 2.12. For an extended discussion on how sinusoidal positional encodings help overcome spectral bias in neural networks please refer to [123].

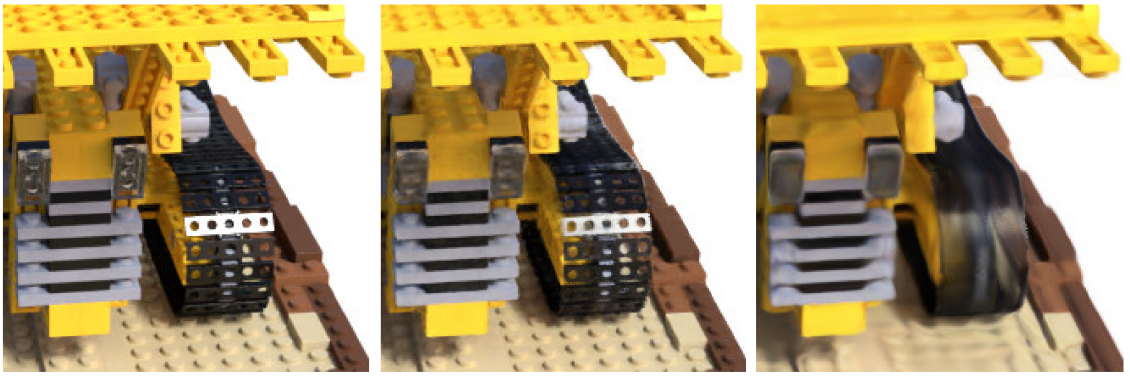


Figure 2.12: **Effect of Positional Encodings in NeRF.** Image Source: Mildenhall *et al*. [83]

As shown in detail in Algorithm 4, the rendering color values associated with ray, involves the following main steps:

- sampling 3D points along the ray,
- querying the MLP model for color and density values of each 3D point,
- converting point densities into interval opacities, and
- calculating the final out color values through back-to-front alpha compositing.

Training During training, m rays are randomly sampled from the training input images and are volume rendered using the algorithm discussed above in Algorithm 4. The MLP is trained by minimizing the mean squared error between the rendered ray colors, $\{\hat{c}_i\}_{i=1}^m$ and their ground-truth color values $\{\hat{c}_i\}_{i=1}^m$, $\mathcal{L} = \frac{1}{m} \sum_{i=1}^{m} ||\hat{c}_i - c_i||_2^2$. The NeRF rendering procedure is trivially differentiable and allows training the MLP networks by minimising the loss term $\mathcal{L}$.

Algorithm 4 NeRF: Rendering Rays

Get uniform sampling locations $\{t\}_{t=1}^{D}$, in the range $[t_{near}\, t_{far}]$
Sample $3D$ points $\{(x_i, y_i, z_i) = \mathbf{o} + t_i\, \mathbf{d}\}_{t=1}^{D}$
Get spherical viewing directions (θ, ϕ) from unit vector $\mathbf{d}$
for $i \in [0, 1, \ldots, N-1]$ **do**
    $\mathbf{c}_i, \sigma_i = \mathbf{F}_\Theta(x_i, y_i, z_i, \theta, \phi)$ ▷ Get point color and density
    $\alpha_i = 1 - \exp(-\sigma_i\,(t_{i+1} - t_i))$ ▷ Convert density to opacity
end for
$\mathbf{c} = \sum_{i=0}^{N-1} c_i\, \alpha_i \prod_{j=0}^{i-1} (1 - \alpha_j)$ ▷ Back-to-front Alpha Composition
return $\mathbf{c}$

3 Deep Stereo Vision Guided View Interpolation

This chapter presents a fast and accurate Novel View Synthesis approach from multi-view stereo images. The proposed method, FastNVS, decomposes the Novel View Synthesis problem into two sub-tasks: *view-dependent geometry* estimation and *texture inpainting*. Both tasks are structured prediction problems that could be effectively learned with CNNs. Experiments on the KITTI Odometry dataset [39] show that FastNVS is more accurate and significantly faster than the current state-of-the-art.

The work presented in this chapter is published in the following article:

1. Tewodros Amberbir **Habtegebrial**, Christian Bailer, Kiran Varanasi and Didier Stricker Fast View Synthesis with Deep Stereo Vision.
   In *14th International Conference on Computer Vision Theory and Applications (VISAPP)*, 2019.

Contents

3.1 Introduction **42**
3.2 Related Work **44**
3.3 Proposed Method **45**
3.4 Experiments **52**
3.5 Conclusion and Future Work **57**

Figure 3.1: **Sample renderings from the KITTI dataset [39]** using our proposed method. Images are rendered using four neighbouring views. From left to right, the median baseline between consecutive input views increases from $0.8m$ to $2.4m$.

3.1 Introduction

Novel View Synthesis (NVS) is defined as the problem of rendering a scene from a previously unseen camera viewpoint, given other reference images of the same scene. This is an inherently under-determined problem due to perspective projection, occlusions in the scene, and the effects of lighting and shadows that vary with a viewpoint. Due to this inherent ambiguity, this can be solved only by learning valid scene priors and hence, is an effective problem to showcase the application of machine learning to computer vision. Accurate rendering of novel views can be a useful component for many computer vision applications, such as pedestrian detection or robot tool positioning, the accuracy of which is significantly affected by the viewpoint. Many applications require extremely robust and fast or even real-time NVS, which is currently beyond the state of the art. Furthermore, NVS is essential for rendering real-world scenes in virtual reality.

In the early 1990s, methods for NVS were proposed to deal with slight viewpoint changes, given images taken from relatively close viewpoints. Then NVS can be performed through view interpolation [21], warping [109] or rendering with stereo reconstruction [107]. A dense matrix of camera views can be used to sample the *plenoptic function* [2], which can be rendered into new viewpoints. Levoy and Hanrahan [73], McMillan and Bishop [81] introduced Image-based Rendering (IBR) as an attempt to reconstruct novel views from these samples of plenoptic function. This framework is not very suitable when the plenoptic function is sampled sparsely, i.e., when the input camera views are separated by a large-baseline. Across wide baselines, NVS is a particularly challenging problem due to sharp foreshortening effects, scale changes and the 3D ro-

tation of objects. In this chapter, we address this challenging problem and demonstrate results on the KITTI dataset, where stereo pairs of images were recorded from a moving car.

There were a few methods proposed over the years to solve the problem of large-baseline NVS [16], [152], [36], [41], [90]. Some methods are based on Structure-from-Motion (SfM) [152], which can produce high-quality novel views in real time [16], but have limitations when the input images contain strong noise, illumination change and highly non-planar structures like vegetation. These methods need to produce depth synthesis for poorly reconstructed areas in SfM, which is challenging for intricate structures. In contrast to these methods, neural networks can be trained end-to-end to render NVS directly [36]. This is the paradigm we follow in this chapter.

Many recent works have addressed different facets of end-to-end training of neural networks for NVS [31], [126], [150], [71], [139]. These methods typically perform well under a restricted scenario, where they have to render geometrically simple scenes, like a single object rendered from the ShapeNet Dataset [15], but addressing real-world scenes with large variations is still challenging. The state-of-the-art large-baseline view synthesis approach that works well under challenging scenarios is *DeepStereo*, introduced by Flynn *et al*. [36]. *DeepStereo* generates high-quality novel views on the KITTI dataset [39], where older SfM-based methods such as [16] do not work at all. This algorithm uses plane-sweep volumes and processes them with a double-tower CNN (with *color* and *selection* towers). However, processing plane-sweep volumes of all reference views jointly imposes very high memory and computational costs. Thus, *DeepStereo* is far slower than previous SfM-based methods such as [16].

In this chapter we propose a novel alternative to *DeepStereo*, which is two orders of magnitude faster. Our method avoids performing expensive calculations on the combination of plane-sweep volumes of reference images. Instead, we predict a proxy scene geometry for the input views with stereo-vision. Using forward-mapping (section 3.3.2), we project input views to the novel view. Forward-mapped images contain a large number of pixels with unknown color values. Rendering of the target view is done by applying texture inpainting on the warped images. Our rendering pipeline is fully-learnable from a sequence of calibrated stereo images. Compared to *DeepStereo*, the proposed approach produces more accurate results while being significantly faster. The main contributions of this chapter are the following:

- We present a Novel View Synthesis approach based on stereo-vision. The proposed approach decomposes the problem into *proxy* geometry prediction and texture inpainting tasks. Every part of our method is fully learnable from input stereo reference images.

- Our approach provides an affordable large-baseline view synthesis solution. The proposed method is faster and even more accurate than the current *state-of-the-art* [36] large-baseline view synthesis. The proposed approach takes seconds to render a single frame while DeepStereo [36] takes minutes.

3.2 Related Work

Image-based Rendering has enjoyed a significant amount of attention from the computer vision and graphics communities. Over the last few decades, several approaches of Image-based Rendering and modelling were introduced [21], [109], [110], [2], [107]. Fitzgibbon *et al*. [34] present a solution that solves view synthesis as texture synthesis by using image-based priors as regularization. Chaurasia *et al*. [16] presented high-quality view synthesis that utilizes 3D reconstruction. Recently, Penner *et al*. [90] presented a view synthesis method that uses soft 3D reconstruction via fast local stereo-matching similar to [55] and occlusion-aware depth-synthesis. Kalantari *et al*. [65] used deep convolutional networks for view synthesis in light fields.

Encoder-decoder Networks: Encoder-decoder models have been used in generating unseen views of objects with simple geometric structure, (e.g. cars, chairs, etc) [126], [31]. However, the renderings generated from encoder-decoder architectures are often blurry. Zhou *et al*. [150] used encoder-decoder networks to predict appearance flow, rather than directly generating the image. Compared to direct novel view generating methods [126], [31], the appearance-flow based method produces crisper results. Nonetheless, the appearance flow based method also fails to produce any convincing results in natural scenes.

Deep Generative Models: As it has already been done in encoder-decoder networks [31], [126], view synthesis could be proposed as a generative modeling task. A generative model of view synthesis would have to hallucinate what a scene looks like from a certain camera pose, given reference views of the scene. Most of the recent generative neural networks such as [61], [133], [20], however, work in scenarios where almost no strong geometric manipulation of the input is needed. For instance, converting images to semantic labels and vice versa. This limits their applicability in Novel View Synthesis.

DeepStereo: Flynn *et al*. [36] proposed the first CNN-based large-baseline Novel View Synthesis approach. DeepStereo has a double-tower (color and selection towers) CNN architecture. DeepStereo takes a volume of images projected using multiple depth planes, known as *plane-sweep volume* as input. The *color-tower* produces renderings for every depth plane separately. The *selection-tower* estimates probabilities for the renderings computed for every depth plane. The output image is then computed as a weighted average of the rendered color-images. DeepStereo generates high-quality novel views from a plane sweep volume generated from a few (typically 4) reference views. To the best of our knowledge, *DeepStereo* is the most accurate large-baseline method, proven to be able to generate accurate novel views of challenging natural scenes.

Depth Prediction-based view synthesis approaches

Supervised depth prediction with CNNs has been widely studied by the computer vision community [32, 142, 72]. Recent works [40, 148] demonstrated that CNN-based monocular depth prediction could be learned even in the absence of ground truth depth. Godard *et al*. [40] and Zhou *et al*. [148] learn to predict depth by using stereo-reconstruction [40] and multi-view reconstruction losses, respectively. In addition to monocular depth prediction, CNN-based methods have been successful in stereo depth prediction tasks [67]. Kendall *et al*. [67] presented a fast and accurate supervised stereo-depth prediction network called Geometry and Context Network (GCNet). Recently, monocular depth prediction-based view synthesis methods have been proposed [40, 148]. Despite being fast, these works produce results with significantly lower quality compared to multi-view approaches such as *DeepStereo* [36].

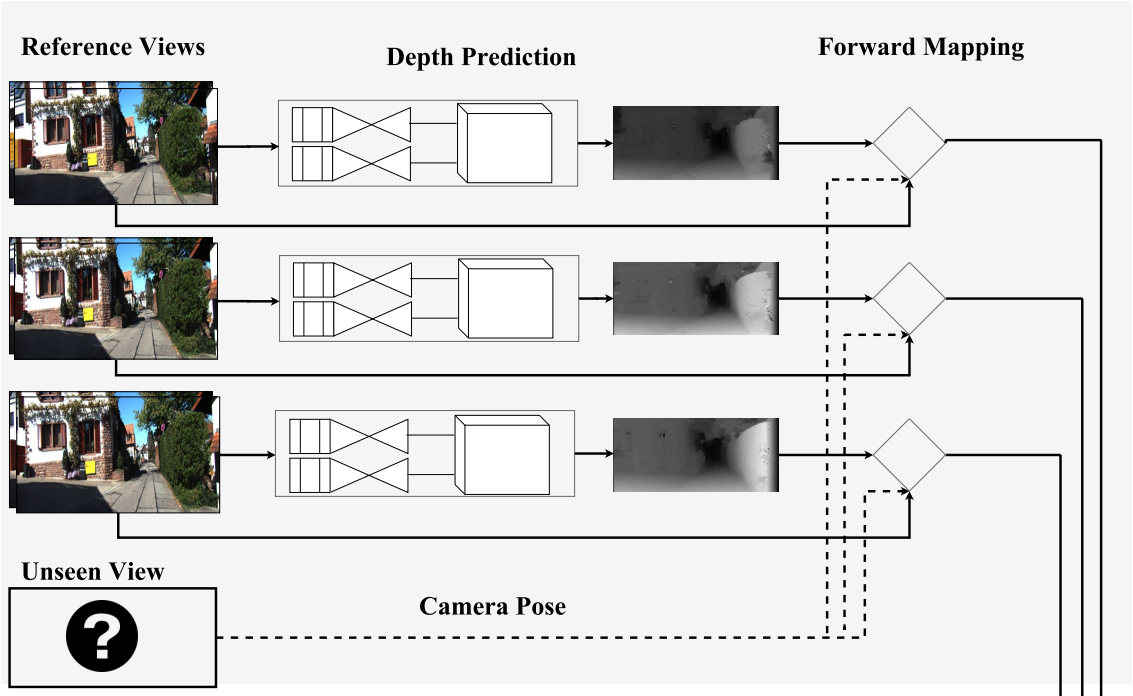
3.3 Proposed Method

In this section we discuss our proposed view synthesis approach called FastNVS. Our aim is to generate the image $\mathcal{X}^t$ of a scene from a novel viewpoint, using a set of reference stereo-pairs $\{\{\mathcal{X}_L^1, X_R^1\}, \{\mathcal{X}_L^2, \mathcal{X}_R^2\}, \ldots, \{\mathcal{X}_L^V, \mathcal{X}_R^V\}\}$[1] and their poses $\{\mathcal{P}^1, \mathcal{P}^2, \ldots, \mathcal{P}^V\}$ w.r.t the target view. The proposed method has three main stages, namely *proxy scene geometry estimation*, *forward-mapping* and *texture inpainting*. As shown in Figure 3.2 the view synthesis is performed as follows: first, a proxy scene geometry is estimated as a dense depth map. The estimated depth map is used to forward-map the input views to the desired novel viewpoint. Forward mapping (described in section 3.3.2) leads to noisy images with a large number of holes. The final rendering is, therefore, generated by applying *texture inpainting* on the forward-mapped images. Both the depth estimation and texture inpainting tasks are learned via convolutional networks. Components of our view synthesis pipeline are trainable using a dataset that contains a sequence of stereo-pairs with known camera poses.

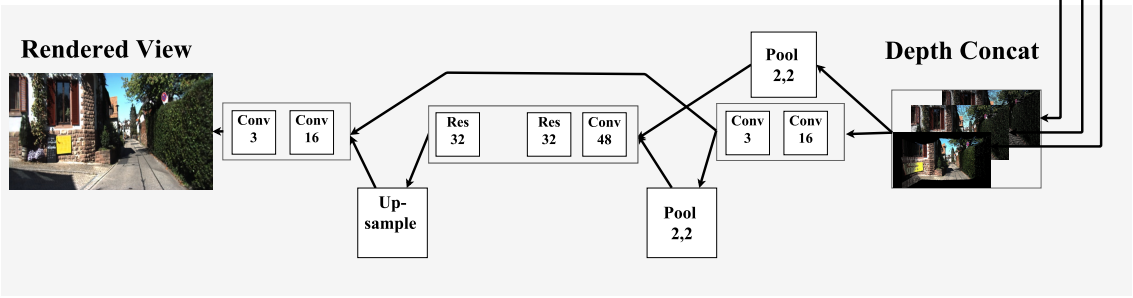
3.3.1 Depth Prediction Network

We train a convolutional network to estimate depth from an input stereo pair. The training set for the depth prediction CNN is generated by sampling M stereo pairs from our training set. The proposed network shown in Figure 3.3, is composed of the following stages: *feature extraction*, *feature-volume aggregation* and *feature-volume filtering*. Architecture-wise our network is similar to GCNet [67]. However, there are several key differences, including the fact that our network is trained in an unsupervised manner. Similar to our depth prediction network, a recent work [147] also investigates learning

[1] subscripts L and R indicate left and right views, respectively



(a) Proxy geometry estimation and view-wapring



(b) Texture in-painting

Figure 3.2: **Overview of FastNVS approach.** In the first stage (a), we begin by estimating dense depth maps from the input reference stereo pairs using an unsupervised stereo-depth prediction network. Estimated depth maps are used to project input views to the target view via Forward-mapping, shown in Equation 3.7. As shown in (b), the output novel view is rendered with the *texture inpainting* applied on the forward mapped views.

stereo disparity prediction without using ground truth data.

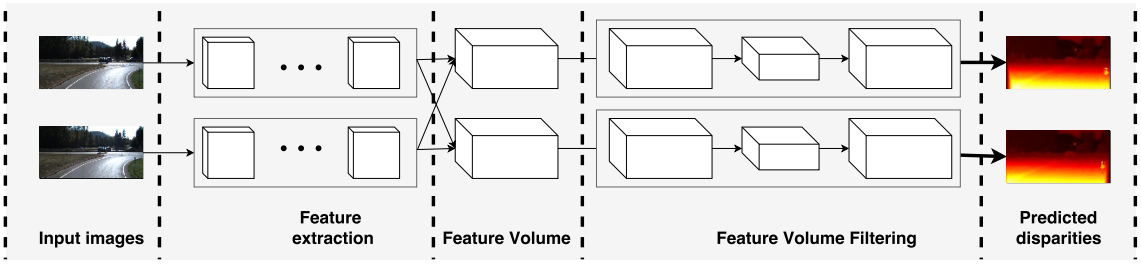


Figure 3.3: **Depth prediction network architecture.**

Feature Extraction:

Generating robust feature descriptors is an important part of stereo matching and optical flow estimation systems. Recently, CNN-based feature generation and matching has been used in stereo matching and optical flow estimation papers [6], [37], [5], [142]. In this chapter we extract image-features using a convolutional network. Applying the fully convolutional network (Table 3.1) on *left* and *right* stereo-images, we extract features $\mathcal{F}_L$ and $\mathcal{F}_R$.

Layers	Kernel size	Stride	Input channels	Output channels	Nonlinearity
conv_0	5x5	2x2	32	32	ReLU
res_1 to res_9	3x3	1x1	32	32	ReLU + BN
conv_10	3x3	1x1	32	32	-

Table 3.1: **Details of the feature extraction stage.** In the first entry of the third row, we use *res_1 to res_9*, as a shorthand notation for a stack of 9 identical residual layers.

Feature-volume Generation:

Features $\mathcal{F}_L$ and $\mathcal{F}_R$ are aggregated into feature-volumes $\mathcal{V}_L$ and $\mathcal{V}_R$ for the left and right images, respectively. Feature volumes are data structures that are convenient for matching features. Feature volume $\mathcal{V}_L$ is created by concatenating $\mathcal{F}_L$ with $\mathcal{F}_R$ translated at different disparity levels $d_i \in \{d_1, d_2, \dots d_D\}$. $\mathcal{V}_L = [(\mathcal{F}_L, \mathcal{F}_R^{d_1}), (\mathcal{F}_L, \mathcal{F}_R^{d_2}), \dots, (\mathcal{F}_L, \mathcal{F}_R^{d_D})]$[2] Similarly, $\mathcal{V}_R$ is created by translating $\mathcal{F}_L$ and concatenating it with $\mathcal{F}_R$.

Feature-volume Filtering:

Generated feature volumes aggregate left and right image features at different disparity levels. This stage is posed with the task of computing pixel-wise probabilities over

[2]We used (X, Y) to denote *concatenating* X and Y across the first dimension and we use [X and Y] to denote *stacking* X and Y, which creates a new first dimension

disparities from feature volumes. Since $\mathcal{V}_L$ and $\mathcal{V}_R$ are composed of features vectors spanning 3-dimensions (disparity, image-height, and image-width) it is convenient to use 3D convolutional layers. 3D convolutional layers are able to utilize neighborhood information across the above three axes. The 3D-convolutional encoder-decoder network used in this chapter is presented in Table 2. All layers, except the last layer (*tr_conv3d_out*) are followed by a ReLU Non-Linearity and Batch Normalization.

Layer	Input	Output size	Stride
conv3d_1	Feature Volume	(D, 32, H/2, W/2)	(1,1,1)
conv3d_2	conv3d_1	(D, 32, H/2, W/2)	(1,1,1)
conv3d_3	conv3d_2	(D, 32, H/4, W/4)	(2,2,2)
conv3d_4	conv3d_3	(D, 32, H/4, W/4)	(1,1,1)
conv3d_5	conv3d_4	(D, 32, H/4, W/4)	(1,1,1)
conv3d_6	conv3d_5	(D, 32, H/8, W/8)	(2,2,2)
conv3d_7	conv3d_6	(D, 32, H/8, W/8)	(1,1,1)
conv3d_8	conv3d_7	(D, 32, H/16, W/16)	(2,2,2)
tr_conv3d_1	conv3d_8	(D, 32, H/8, W/8)	(2,2,2)
tr_conv3d_2	conv3d_7, tr_conv3d_1	(D, 32, H/4, W/4)	(2,2,2)
tr_conv3d_3	conv3d_5, tr_conv3d_2	(D, 32, H/2, W/2)	(2,2,2)
tr_conv3d_4	conv3d_2,tr_conv3d_3	(D, 32, H, W)	(2,2,2)
output	tr_conv3d_4	(D, 1, H, W)	(1,1,1)

Table 3.2: **Feature volume filtering network.** Details of the 3D convolutional encoder-decoder network used for feature volume filtering. Layers with "conv3d_" are 3d convolutional layers while names that start with "tr_conv3d_" represent 3d transposed convolution.

Let's denote the output of applying feature-volume filtering on $\mathcal{V}_L$ and $\mathcal{V}_R$ as $\mathcal{C}_L$ and $\mathcal{C}_R$, respectively. $\mathcal{C}_L$ and $\mathcal{C}_R$, are tensors represent pixel-wise disparity "confidences". In order to convert confidences into pixel-wise disparity maps, a *soft arg-min* function is used. $\mathcal{C}_L$ and $\mathcal{C}_R$ represent negative of confidence, hence, we use *soft-argmin*, instead of *soft-argmax*. The *soft-argmin* operation (Equation 3.1) is a differentiable alternative to *argmin* [67]. For every pixel location, *soft-argmin* first normalizes the depth confidences into a proper probability distribution function. Then, disparities are computed as the expectation of the disparity under the normalized distributions. Thus applying *soft-min* on $\mathcal{C}_L$ and $\mathcal{C}_R$ gives disparity maps $\mathcal{D}_L$ and $\mathcal{D}_R$, respectively.

$$\begin{aligned} \mathcal{P}_L(i,x,y) &= \frac{e^{-\mathcal{C}_L(i,x,y)}}{\sum_{j=1}^{D} e^{-\mathcal{C}_L(j,x,y)}} \\ \mathcal{D}_L(x,y) &= \sum_{i=1}^{D} disp(i) * \mathcal{P}_L(i,x,y) \end{aligned} \tag{3.1}$$

The estimated disparities $\mathcal{D}_L, \mathcal{D}_R$ are disparities that encode motions of pixels between the left and right images. We convert the predicted disparities into a sampling grid in order to warp left image to the right image (and vice versa). Once sampling grid is created we apply bi-linear sampling [63] to perform the warping. Denoting the bi-linear sampling operation as Φ, the warped left and right images could be expressed as $\tilde{\mathcal{X}}_L = \Phi(\mathcal{X}_R, \mathcal{D}_L)$ and $\tilde{\mathcal{X}}_R = \Phi(\mathcal{X}_L, \mathcal{D}_R)$, respectively.

Disparity Estimation Network Training Objective

Our depth prediction network is trained in unsupervised manner by minimizing a loss term which mainly depends on the photometric discrepancy between the input images $\{\mathcal{X}_L,\mathcal{X}_R\}$ and their respective re-renderings $\{\tilde{\mathcal{X}}_L,\tilde{\mathcal{X}}_R\}$. Our network minimizes, the loss term $\mathcal{L}_T$ which has three components: a photometric discrepancy term $\mathcal{L}_P$, smoothness term $\mathcal{L}_S$ and left-right consistency term $\mathcal{L}_{LR}$, with different weighting parameters $\lambda_0, \lambda_1,$ and λ_2.

$$\mathcal{L}_T = \lambda_0 \mathcal{L}_P + \lambda_1 \mathcal{L}_{LR} + \lambda_2 \mathcal{L}_S \tag{3.2}$$

Photometric discrepancy term $\mathcal{L}_P$ is a sum of an $L1$ loss and a structural dissimilarity term based on Structural Similarity Index Measure (SSIM) [134] with multiple window sizes. In our experiments we use $S = \{3, 5, 7\}$. N in Equation 3.3 is the total number of pixels.

$$\begin{aligned} \mathcal{L}_P = & \frac{\lambda_1^p}{N}(\|\tilde{\mathcal{X}}_L - \mathcal{X}_L\|_1 + \|\tilde{\mathcal{X}}_R - \mathcal{X}_R\|_1) + \\ & \frac{\lambda_s^p}{N} \sum_{s \in S} (SSIM_s(\tilde{\mathcal{X}}_L, \mathcal{X}_L) + SSIM_s(\tilde{\mathcal{X}}_R, \mathcal{X}_R)) \end{aligned} \tag{3.3}$$

Depth prediction from photo-consistency is an ill-posed inverse problem, where there are a large number of photo-consistent geometric structures for a given stereo pair. Imposing regularization terms encourages the predictions to be closer to the physically valid solutions. Therefore, we use an edge-aware smoothness regularization term $\mathcal{L}_\mathcal{S}$ in Equation 3.5 and left-right consistency loss [40] Equation 3.4. $\mathcal{L}_{LR}$ is computed by warping disparities and comparing them to the original disparities predicted by the network.

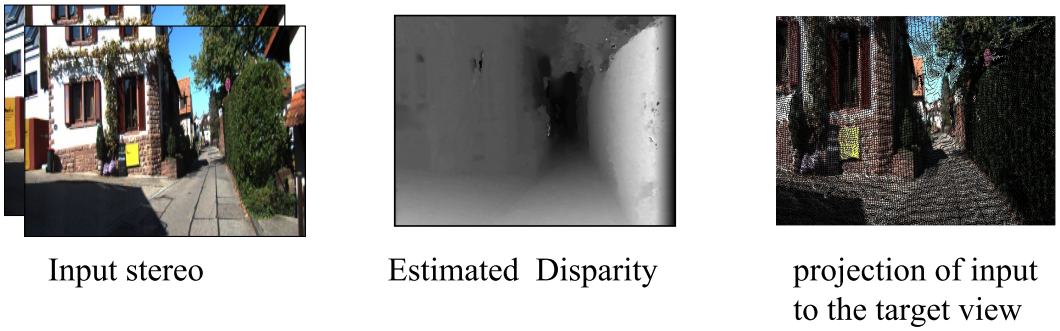
$$\begin{aligned} & \mathcal{L}_{\mathcal{LR}} = \|\tilde{\mathcal{D}_\mathcal{L}} - \mathcal{D}_\mathcal{L}\|_1 + \|\tilde{D_R} - \mathcal{D}_R\|_1 \\ & \text{where } \tilde{\mathcal{D}_\mathcal{L}} = \Phi(\mathcal{D}_R, \mathcal{D}_L) \text{ and } \tilde{\mathcal{D}_\mathcal{R}} = \Phi(\mathcal{D}_L, \mathcal{D}_R) \end{aligned} \tag{3.4}$$

Smoothness term $\mathcal{L}_\mathcal{S}$ forces disparities $\mathcal{D}_\mathcal{L}$ and $\mathcal{D}_\mathcal{R}$ to have small gradient magnitudes in smooth image regions. However, the term allows large disparity gradients in regions where there are strong image gradients.

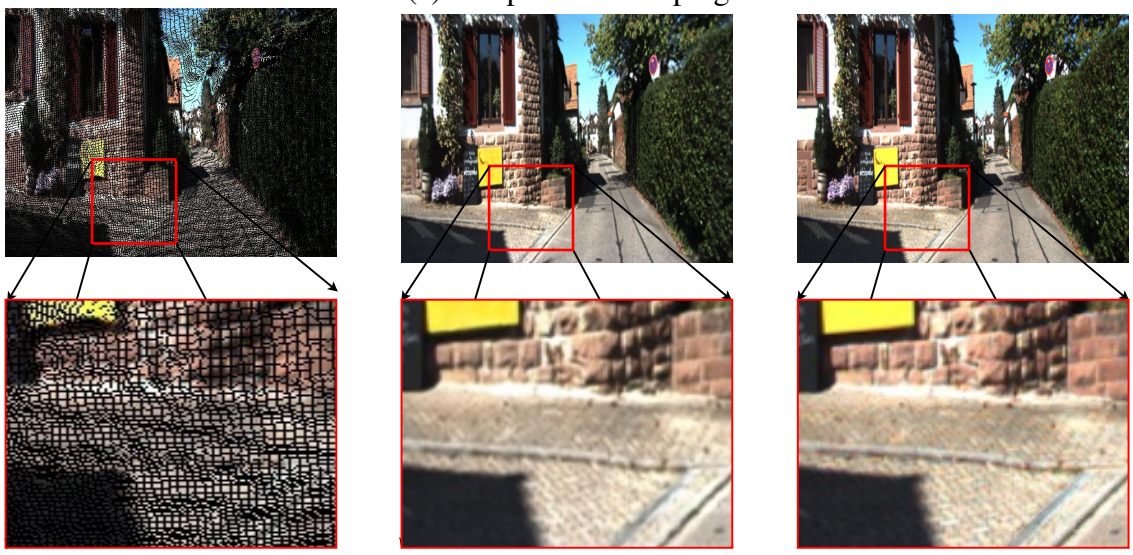
$$\begin{aligned} \mathcal{L}_\mathcal{S} = & \nabla_x \mathcal{D}_L e^{(-\nabla_x \mathcal{X}_L)} + \nabla_y \mathcal{D}_L e^{(-\nabla_y \mathcal{X}_L)} + \\ & \nabla_x \mathcal{D}_R e^{(-\nabla_x \mathcal{X}_R)} + \nabla_y \mathcal{D}_R e^{(-\nabla_y \mathcal{X}_R)} \end{aligned}_3 \tag{3.5}$$

Bilinear sampling, Φ allows back-propagation of error (sub-)gradients from the output such as $\tilde{\mathcal{X}}s$ back to the input images $\mathcal{X}$s and disparities $\mathcal{D}$s. Therefore, it is possible to use standard back-propagation to train our network. Formal derivations for the back-propagation of gradients via a bilinear sampling module could be found in [63].

3.3.2 Forward Mapping



(a) Sample view-warping



(b) Close-up view of warped and target images

Figure 3.4: **View-warping with forward mapping.**

We apply our trained stereo depth predictor on the $\mathcal{V}$ input stereo pairs $\{\{\mathcal{X}_L^1, X_R^1\}, \{\mathcal{X}_L^2, \mathcal{X}_R^2\}, \ldots, \{\mathcal{X}_L^V, \mathcal{X}_R^V\}\}$ and generate disparities for the left camera input views, $\{\mathcal{D}_L^1, \mathcal{D}_L^2, \ldots \mathcal{D}_L^V\}$. The right stereo views are used only for estimating disparities.

[3] ∇_x is gradient w.r.t x, similarly ∇_y is gradient w.r.t y

Forward mapping and texture inpainting are done only on the images from the left camera. In this section, unless specified otherwise, we refer to the images from left camera as the input images/views and we drop the subscripts L and R.

Forward-mapping projects input views to the target-view using their respective depth-maps. The predicted disparities of the input views could be converted to depth values. Depth $\mathcal{Z}^i_{w,h}$ for a pixel at location (w, h) in the $i-th$ input view, can be computed from the corresponding disparity $\mathcal{D}^i_{w,h}$, as follows:

$$\mathcal{Z}^i_{w,h} = \frac{f_x * B}{\mathcal{D}^i_{w,h}} \tag{3.6}$$

where K is the intrinsic camera matrix, B is the baseline, and f_x is focal length.

The goal of forward mapping is to project the input views to the target view, t. Given the relative pose between the input-view i and target-view as a transformation matrix $P^i = [R^i|T^i]$, pixel $p^i_{h,w}$ (pixel location $\{h, w\}$ in view i) will be forward mapped as follows, to a pixel location $p^t_{x,y}$ on the target view:

$$\begin{aligned} &[x', y', z'] \sim K P^i Z^i_{h,w} K^{-1} [h, w, 1]^T \\ &x = \lfloor x'/z \rfloor \text{ and } y = \lfloor y'/z \rfloor \end{aligned} \tag{3.7}$$

Following a standard forward projection Equation 3.7, the reference input frames $\{\mathcal{X}^1, \mathcal{X}^2, \ldots, \mathcal{X}^V\}$ are warped to the target view $\{\mathcal{W}^1, \mathcal{W}^2, \ldots, \mathcal{W}^V\}$. As shown in Figure 3.4, forward-mapped views have a large number of pixel locations with unknown color values. These holes are created for various reasons. First, forward-mapping is a one-to-one mapping of pixels from the input views to the target view. This creates holes as it does not account for zooming in effects of camera movements, which could lead to one-to-many mapping. Moreover, occlusion and rounding effects lead to more holes. In addition to holes, some warped pixels have wrong color values due to inaccuracies in depth prediction.

3.3.3 Texture Inpainting

The goal of our texture inpainting network is to learn generating the target view $\mathcal{X}^t$ from the set of warped input views $\{\mathcal{W}^1, \mathcal{W}^2, \ldots, \mathcal{W}^V\}$. This is a structured prediction task where the input and output images are aligned. Due to the effects mentioned above in section 3.3.2, texture mapping results in noisy warped views, see Figure 3.4. Forward-mapped images $\mathcal{W}^i s$ contain two kinds of pixels: *noisy-pixels*, those with unknown (or wrong) color values and *good-pixels*, those with correct color value. Ideally we would like to hallucinate the correct color value for the noisy pixels while maintaining the *good* pixels.

The architecture of our proposed *inpainting* network is inspired by Densely Connected [57] and Residual [47] network architectures. Details of the network architecture are

presented in Table 3.3. The network has residual layers with long range skip-connections that feed features from early layers to the top layers. The architecture design used on our network is designed to facilitate flow of activations as well as gradients through the network. The texture inpainting network is trained by minimizing L1 loss between the predicted novel views and the original images.

Block	**Layer**	**Input**	**Input,Out. channels**	**Output size**
Block 0	conv_0	warped views	4*4,32	H, W
Block 0	res_1	conv_0	32,32	H, W
Block 1	res_2	pool(res_1), pool(warped views)	32+16, 48	H/2, W/2
Block 1	res_3	res_2	48 ,48	H/2, W/2
Block 1	res_4	res_3	48 ,48	H/2, W/2
Block 1	res_5	res_4	48 ,48	H/2, W/2
Block 1	res_6	res_5	48 ,48	H/2, W/2
Block 1	res_7	res_6	48 ,48	H/2, W/2
Block 1	res_8	res_7	48 ,48	H/2, W/2
Block 2	conv_9	upsample(res_8), res_1	48+32,48	H, W
Block 2	res_10	conv_9	48, 48	H, W
Block 2	res_11	res_10	48, 48	H,W
Block 3	conv_12	res_11	48, 16	H,W
Block 3	output	conv_12	16, 3	H,W

Table 3.3: **Texture inpainting network architecture.** The network is mainly residual, except special convolution layers which could be used as input and output layers. Similar to *Densely Connected Nets* [57], at the beginning of each block, the number of channels in the concatenated feature maps could be decreased via a convolutional layer.

3.4 Experiments

We tested our proposed approach on the KITTI [39] public computer vision benchmark. Our approach has lower rendering error than the current state-of-the-art [36] (see Table 3.4). Qualitative evaluation also shows that our method better preserves the geometric details of the scene, shown in Figure 3.5.

Network Training and Hyper-parameters

The depth predictor loss term has three weighting parameters: λ_0 for photometric loss, λ_1 for left-right consistency loss and λ_2 for local smoothness regularization. The weighting parameters have to be set properly, as some combinations of weights might lead trivial

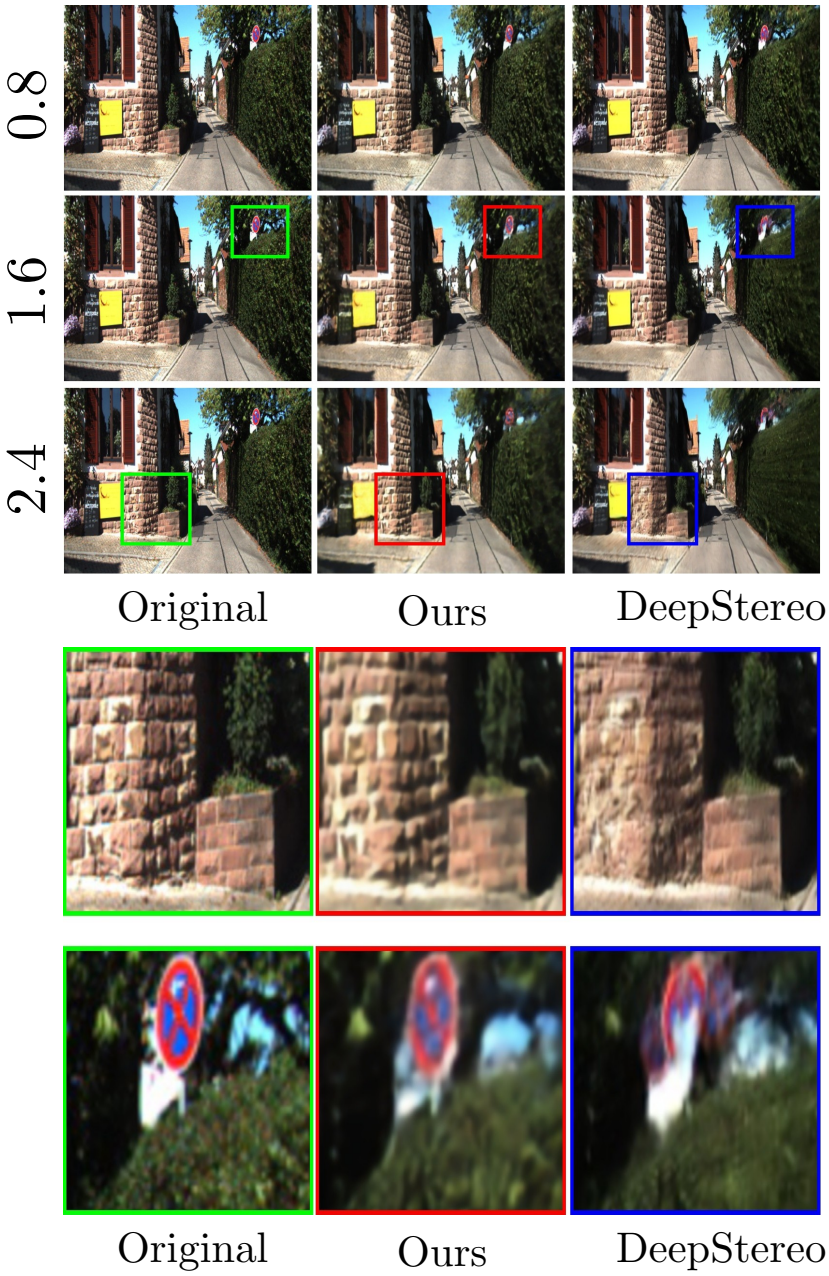


Figure 3.5: **Visual comparison against DeepStereo [36].** Close-up views show that our FastNVS method preserves the geometric structure better and *DeepStereo* has ghosting where the traffic sign is replicated multiple times. Our rendering resembles the target except for a slight amount of blur. The median baseline between input views is shown to the left of each novel view. As expected higher baselines lead to stronger rendering artifacts.

solutions such as a constant disparity output for every pixel. The weighting parameters used in our experiments are the following: $\lambda_0 = 5$, $\lambda_1 = 0.01$, $\lambda_2 = 0.0005$. The photometric loss is also a weighted combination of $L1$ and SSIM losses with different window sizes. These weighting factors are set to be $\lambda_P^1 = 0.2$, $\lambda_P^3 = 0.8$, $\lambda_P^5 = 0.2$ and $\lambda_P^7 = 0.2$. As discussed in section 3. the depth-prediction and texture inpainting networks are trained separately. We train the networks with back-propagation using Adam optimizer [69], with mini-batch size 1 and learning rate of 0.0004. The depth prediction network is trained for 200,000 iterations and the inpainting network is trained for 1 Million iterations.

Dataset:

We evaluate our proposed method on the publicly available KITTI Odometry Dataset [39]. KITTI is an interesting dataset for large-baseline Novel View Synthesis. The dataset has a large number of frames recorded in outdoors sunny urban environment. The dataset contains variations in the scene structure (vegetation, buildings, etc), strong illumination changes and noise which makes KITTI a challenging benchmark. The dataset contains around 23000 stereo pairs divided into 11 sequences (*00* to *10*). Each sequence contains a set of images captured by a stereo camera, mounted on top of car driving through a city. The stereo camera captures frames every $\approx 80\ cms$. Each sequence corresponds to a single drive and extrinsic camera calibrations are available for each sequence separately.

In order to make our results comparable to *DeepStereo* [36], we hold out *sequence 04* for validation, *sequence 10* for test and used the rest for training. In our experiments, 5 consecutive images are used, with the middle one held out as target and the other 4 images are used as reference. Tests are done at different values of spacing between consecutive images. Taking consecutive frames gives spacing of around $0.8\ m$. Sampling every other frame gives a spacing of $\approx 1.6m$ and every third frame gives $\approx 2.4m$ spacing. Numerical rendering accuracy is measured using a mean absolute brightness error, computed per pixel per color channel.

Spacing	Method	Test 0.8 m	Test 1.6 m	Test 2.4 m
Train 0.8 m	Ours	**6.66**	**8.90**	**12.14**
	DeepStereo	7.49	10.41	13.44
Train 1.6 m	Ours	**6.92**	**8.47**	**10.38**
	DeepStereo	7.60	10.28	12.97
Train 2.4 m	Ours	**7.47**	**8.73**	**10.28**
	DeepStereo	8.06	10.73	13.37

Table 3.4: **Quantitative evaluation of our method against *DeepStereo*.** We used our texture inpainting network with residual blocks as shown in Table 3.3. Our approach outperforms *DeepStereo* [36] in all cases. Each row shows the performance of a method that is trained on a specific input camera spacing and tested on all three spacings.

Results:

Table 3.4 shows the results of our proposed approach compared to DeepStereo [36], on the KITTI dataset. Our approach outperforms *DeepStereo* in all spacings of the input views, while being significantly faster. In Figure 3.5, sample renderings are shown for a qualitative evaluation of our approach and *DeepStereo*. As shown in the lowest two rows of Figure 3.5, renderings of *DeepStereo* suffer from ghosting effects due to cross-talk between different depth planes, which led to the replicated traffic sign and noisy bricks (as could be seen in the close-up views).

We performed tests to compare our texture inpainting architecture against the UNet architecture [102]. Furthermore, in order to evaluate the significance of the texture inpainting stage, we measured the accuracy of our system when the texture inpainting stage is replaced by simple median filtering scheme applied on the warped views. Table 3.5 shows that our inpainting network (with residual layers) achieves better performance than UNet and the "basic" median filtering baseline. A test performed to investigate the effect of using residual layers instead of convolutional layers shows that residual layers give better performance.

Method	Test 0.8 m	Test 1.6 m	Test 2.4 m
Ours with Residual blocks	**6.66**	**8.47**	**10.28**
Ours with Conv blocks	6.70	8.57	10.34
UNet	7.08	9.60	11.17
Median of Warped-views	18.97	31.72	41.29

Table 3.5: **Testing different choices of texture inpainting networks.** We test the effect of using residual and convolutional layers. We also compare the performance of our network against the UNet [102] architecture and the median of warped input views. In all baselines, our network with residual layers gives the lowest error.

Increasing the input baseline: As shown in Tables 3.4 and 3.5, increasing the baseline between input views makes the view synthesis problem harder. The reported numerical accuracy of both FastNVS and DeepStereo drops as baseline increases. In Figures 3.6, and 3.7, we compare Novel View Synthesis results at baselines $80\ cms$ in Figure 3.6 and $160\ cms$ in Figure 3.7. As we increase the baseline more artifacts start to emerge in both approaches, however our FastNVS method keeps most of the scene geometry intact.

In Figure 3.4 we compared more results that visually compare our method against the state-of-the-art method DeepStereo [36].

Figure 3.6: **Novel views from** $80\,cm$ **baseline inputs** Results from FastNVS and DeepStereo [36] on test scene from the KITTI dataset. The input images are sampled at approximately $80cms$ baseline between them. At this relatively small input baseline, both approaches perform very well.

Figure 3.7: **Novel views from** $160\,cm$ **baseline inputs** Results from FastNVS and DeepStereo [36] on test scene from the KITTI dataset. The input images are sampled at approximately $160cms$ baseline between them.

Timing

During the test phase, at a resolution of 528×384, our depth prediction stage and the forward mapping take 6.08 seconds and 2.60 seconds (for four input neighbouring views), respectively. The texture rendering takes 0.05 seconds. Thus, the total time adds up to 8.73 seconds, per frame. This is much faster than *DeepStereo* which takes 12 minutes to render a single frame of resolution 500×500. Our experiments are performed on a multi-core CPU machine with a single Tesla V100 GPU.

3.5 Conclusion and Future Work

We presented a fast and accurate Novel View Synthesis pipeline based on stereo-vision and convolutional networks. Our method decomposes Novel View Synthesis into view-dependent geometry estimation and texture inpainting problems. Thus, our method utilizes the power of convolutional neural networks in learning structured prediction tasks. Our proposed method is tested on a challenging benchmark, where most existing approaches are not able to produce reasonable results. The proposed approach is significantly faster and more accurate than the current state-of-the-art.

4 Spherical Novel View Synthesis with Soft Occlusion Multisphere Images

In the previous chapter, we introduced a fast and accurate Novel View Synthesis method dubbed *FastNVS*. Despite its advantages, FastNVS is limited to multi-view stereo imagery which normally offers only a limited field of view. Approaches such as FastNVS are not suitable for processing spherical images as they are not invariant to the distortions present in spherical images. This chapter presents a technique for Spherical Novel View Synthesis (SNVS) that estimates 360° views at arbitrary camera poses, given a sparse set of spherical input images.

The work presented in this chapter has been published in the following article:

1. Tewodros Amberbir **Habtegebrial**, Christiano Gava, Marcel Rogge, Didier Stricker and Varun Jampani
   SOMSI: Spherical Novel View Synthesis with Soft Occlusion Multi-Sphere Images.
   In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Project page `https://tedyhabtegebrial.github.io/somsi`.

Contents

4.1 Introduction **60**
4.2 Related Work **62**
4.3 Preliminaries **63**
4.4 Approach **64**
4.5 Experiments **69**
4.6 Network Architecture and Training Details **76**
4.7 Calibrating Spherical Light Fields through Perspective Cut-outs . . **77**
4.8 Conclusion **78**

4.1 Introduction

The advent of low-cost 360° imaging devices makes spherical images a standard choice of representation for 3D scenes in comparison to more expensive 3D modelling with artists or depth sensors. Spherical images are widely used to capture and visualize 360° views of scenes with several applications in virtual tourism, navigation, advertisement, etc. However, a spherical image alone delivers a limited viewing experience with only rotations around the center. User navigation (translation) is usually enabled by capturing multiple spherical images thereby allowing the user to hop from one sphere to another.

A vast majority of the existing literature including our FastNVS method introduced in the previous chapter provide techniques tailored for input multi-view perspective images. In order to leverage the advantages of spherical images, this chapter introduces a novel Spherical Novel View Synthesis method. Spherical Novel View Synthesis (SNVS) is the task of estimating in-between 360° views making it possible for seamless continuous user navigation in a scene. See Figure 4.1 (left) for a sample illustration of the problem setting.

A practical SNVS system would have the following properties:

1. *High-quality* synthesis of disoccluded content and view-dependent effects in novel views,

2. *Fast synthesis time* enabling real-time user navigation in a scene and,

3. *Low memory* consumption to run SNVS on mobile hardware such as VR headsets.

Satisfying all these properties is quite challenging. Current SNVS techniques are based on Multisphere Images (MSI)[1] representation [4, 11]. MSI can be seen as a spherical extension of Multiplane Images (MPI) [149, 121], which are widely adopted in view synthesis of common perspective images.

More specifically, MSIs represent a scene as a set of textured spheres centered around a reference point. A key advantage of using MSIs is that rendering spheres is extremely efficient with standard rendering softwares. The simplicity of the rendering and seamless integration with graphics software makes MSIs an appealing choice for real-time rendering applications. On the other hand, current MSI-based techniques such as MatryODShka (MODS) [4] suffer from unsatisfactory quality in synthesized novel views.

The use of Coordinate Multi-Layer Perceptrons (CMLP) is revolutionizing the field of Novel View Synthesis with very high-quality results such as in NeRF [83]. A key drawback of CMLP based techniques is the requirement of large number of training views as well as slow rendering. Several very recent works try to improve NeRF-like techniques in different aspects: improving rendering speed [99, 38, 49], modelling reflectance properties [10], working with in-the-wild images [80], re-lighting [118], generalizing across scenes [141], etc. Even though one could adapt a NeRF-like technique for SNVS task, rendering novel 360° views would be prohibitively slow. Several concurrent techniques to improve

[1] MSI-based SNVS techniques are discussed in Chapter 2

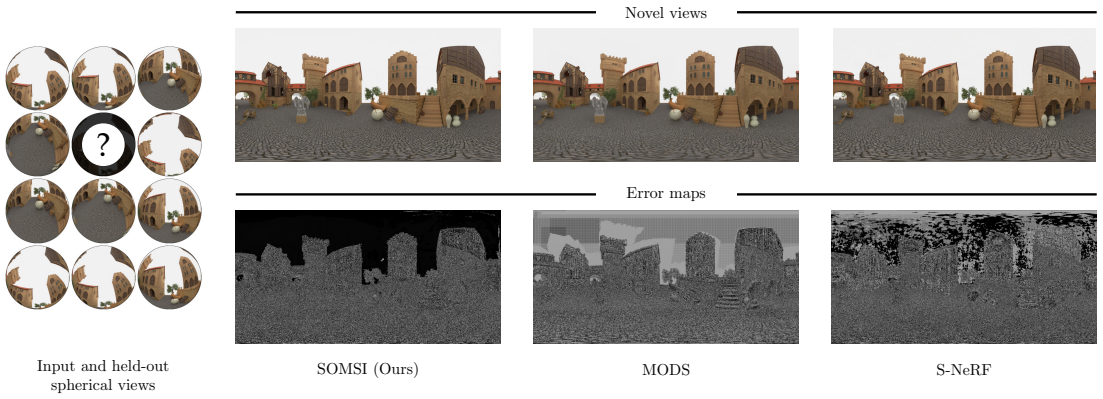


Figure 4.1: **High quality view synthesis with SOMSI.** (Left) Illustration of sample input and held-out target spherical views of a scene. (right) Synthesized novel views and the corresponding error maps for SOMSI (Ours), MODS [4] and S-NeRF [83].

rendering speed of NeRFs [99, 38, 49] are either specific to perspective images or did not demonstrate their use for spherical images in the SNVS task.

In this chapter, we propose a novel SNVS technique that provides a good trade-off between different favourable properties: high-quality view synthesis and fast runtime with low memory requirements. Following [4], we also make use of MSI representation for fast rendering. [4] learns a CNN that takes spherical images as input and produces an MSI representation where each point in each layered sphere has an RGB color and an alpha value associated with it. Several recent MPI works [76, 3, 136] on view synthesis demonstrate the use of high-dimensional learned features in MPI planes instead of color MPIs. Following these approaches and coordinate MLPs, we propose to learn high-dimensional appearance features in MSI using coordinate MLPs. In essence, we combine the strengths of coordinate MLPs with MSI representation, thereby achieving both high quality and fast runtime (rendering).

A key issue in representing high-dimensional features at each point in MSI is high memory consumption. For instance, representing f-dimensional features in d (usually > 64) spheres with m points each leads to a memory complexity of $\mathcal{O}(m \times d \times f)$. Since the number of points m in a 360° spherical image is usually high, the memory complexity will be prohibitive if we represent high-dimensional features in each MSI sphere. We argue that representing dense features in MSI spheres is superfluous as much of the 3D space is empty. As a remedy, we propose a novel MSI scene representation where scene appearance and geometry are factored into two separate data structures. We call this novel MSI representation Soft Occlusion Multisphere Images (SOMSI). SOMSI represents the scene geometry with standard MSI data structure, while the scene appearance is represented as a set of layered 2D feature maps. The key in SOMSI is to represent appearance feature maps using a smaller set of scene-specific occlusion levels instead of a

large set of predefined MSI spheres. In essence, SOMSI representation takes $\mathcal{O}(m \times d \times k)$ memory for scene geometry with soft occlusion masks and $\mathcal{O}(m \times k \times f)$ memory to represent scene appearance features, where k denotes the number of occlusion levels. This strategy scales much better in terms of memory with increasing feature dimension as the number of occlusion levels are significantly lower ($k = 3$ in our case) compared to spheres in MSI (usually $d > 64$). We also propose a novel SOMSI rendering formulation that allows the fast rendering of novel views like with standard MSI representation.

We demonstrate the effectiveness of our SOMSI technique with results on both synthetic and real-world scenes. Figure 4.1 shows sample input and held-out spherical views along with sample results of different techniques. Results show that our approach can considerably outperform previous MSI-based techniques [4]. Our approach can produce high quality novel views that are on par with spherical adaptation of NeRF [83] technique (S-NeRF) while being 2 orders of magnitude faster, with rendering time close to MSI techniques [4]. We make the following contributions:

- We propose a novel Soft Occlusion Multisphere Images (SOMSI) representation that can effectively scale to encode high-dimensional scene appearance features in MSI representation using learnable occlusion layers.
- We propose an efficient way to render novel views from the learned SOMSI representation.
- Our approach effectively combines the advantages of different techniques with high quality view synthesis that is on par with implicit volumetric representations [83] while having fast runtime like with standard MSI representations [4].

4.2 Related Work

View synthesis research has a long history in computer vision and graphics starting from the seminal work of image-space morphing in Chen and Williams [22], light field rendering [73], Lumigraphs [43, 12] followed by multi-view stereo reconstruction techniques [33, 41, 17, 16, 48]. Here, we briefly review the relevant learning-based techniques.
Novel View Synthesis. One of the earliest learning-based Image-based Rendering (IBR) techniques is the DeepStereo method by Flynn *et al*. [36]. DeepStereo trains a CNN to produce novel views from input plane-sweep volumes. An important milestone for the learning-based IBR research was the reintroduction of the Multiplane Images (MPI) by Zhou *et al*. [149]. The view synthesis capabilities of MPIs have been pushed ever further by the DeepView [35] technique. DeepView combines the MPI scene representation with a learned gradient descent based optimization to render highly accurate novel views of challenging real world scenes. A recent extension of MPIs model view-dependent effects [137]. Recently, Neural Radiance Fields (NeRF) [83] techniques uses Coordinate Multi-layer Perceptrons to model a given scene resulting in very high-quality view synthesis results. NeRF renders a scene by performing standard volumetric rendering. However, the

visibility and color information for every point in the rendering volume are determined by invoking the trained MLP which is time-consuming.

Spherical View Synthesis. Single spherical images or panoramic stereo cannot provide parallax as head movement (translation) is not possible. Rendering panoramic scenes with motion parallax is studied in several works [112, 79, 8]. Synthesizing novel views "on demand" is key to enhance user experience by allowing for head movement. Broxton *et al*. [11] presented a light-weight immersive light field video rendering technique by extending MPI formulation of DeepView [35] into a Multisphere Images (MSI) representation. Moreover, a sparse set of MSI spheres are used in order to create a light-weight layered mesh representation that can be rendered on mobile and web platforms. Concurrently with Broxton *et al*. [11], the MODS [4] technique showed the effectiveness of MSIs for 360° view synthesis. MSIs are an attractive option for SNVS as they support real-time rendering with standard rendering softwares.

In this chapter we build upon the MSI scene representation and propose a novel MSI representation that can more efficiently model high-dimensional appearance features.

4.3 Preliminaries

Multisphere Images (MSI)

This is the most commonly used representation for SNVS [11, 4]. MSI can be seen as a spherical extension of Multiplane Images (MPI) that are widely used in view synthesis literature [121, 149]. An MSI is composed of a set of concentric $RGB\alpha$ spheres. The use of spheres to represent a scene allows real-time rendering and easy integration with common rendering softwares such as Unity3D [44] and Blender [27]. This makes MSIs highly suitable for downstream VR applications.

Formally, an MSI is a set of concentric spheres of radii $\{r_i\}_{i=1}^{d}$ with each sphere representing a spherical image. The r_i values are set by linearly sampling the inverse depth range between predefined near r_{near} and far r_{far} values. Each of the d spheres in an MSI has opacity $\alpha \in [0,1]^{m\times 1}$ and color $C \in [0,255]^{m\times 3}$, where $m = h \times w$ is the pixel resolution of the spherical images and d is the number of spheres. MSI color and opacity are represented as a 2D image using the Equi-rectangular Projection (ERP) projection of a sphere onto a plane. The number of spheres in an MSI controls the trade-off between the fidelity of the scene representation and the computational costs (rendering time and memory). Increasing d yields higher fidelity but requires more memory and results in slower rendering.

MSI Rendering

Figure 4.3 shows an illustration of MSI spheres centered around reference view $\mathbf{r}$. Suppose we want to render a spherical image from a novel view center $\mathbf{t}$, we shoot rays from the target center and alpha composite the colors along the intersection points w.r.t. reference-

view MSI spheres. Specifically, for a sample ray direction (corresponding to a location $\mathbf{p} = [u, v]^T$ in the target ERP), we first compute the ray intersections $\mathbf{x}_i^p \in \mathbb{R}^3; i \in \{1, ..., d\}$ with the d MSI spheres using a standard sphere-ray intersection technique [91]. We then record MSI opacity and color values at these intersection points: $\{(\alpha_i^p, C_i^p)\}_{i=1}^d$. The final color C^p for target ray $\mathbf{p}$ is calculated by over-compositing C_i^ps with α_i^ps in a back-to-front manner (also known as alpha composition):

$$C^p = \sum_{i=1}^{d} \alpha_i^p \, C_i^p \prod_{j<i} (1 - \alpha_j^p). \tag{4.1}$$

The same process is used for all the ray directions in the target view to obtain a final novel view spherical image at the target location. Since this process is trivially parallelizable, it is straightforward to use GPUs for fast rendering of novel views. One could also export MSI spheres as textured meshes to leverage standard rendering engines for real-time novel view rendering.

4.4 Approach

Problem

We tackle the problem of Unstructured Spherical Light-Field Interpolation also referred to as Spherical View Synthesis: Given a set of 360° spherical images captured at different locations in a scene, the aim is to estimate spherical images from novel camera viewpoints. Formally, the input to our method is a set of n spherical images, $I_i \in \mathbb{R}^{m \times 3}; i \in \{1, ..., n\}$ each with m pixels, and their $SE(3)$ camera poses, $P \in \mathbb{R}^{n \times 3 \times 4}$.

Given a set of posed spherical images, we learn a scene representation from which one can dynamically render novel views from target camera poses.

Figure 4.2 illustrates the overview of our Spherical Novel View Synthesis (SNVS) technique. We consider one of the input camera poses as reference and optimize a coordinate MLP network that learns to estimate a novel scene representation called 'Soft Occlusion Multisphere Images' (SOMSI) at that reference location. We can then use this representation to dynamically render novel views from this scene representation at the target locations. Next, we describe the SOMSI representation and how we can render novel views from that representation.

4.4.1 Soft Occlusion Multisphere Images (SOMSI)

Motivation

Even though standard MSI representation is efficient in terms of rendering speed, it is limited to low-dimensional RGB appearance in each MSI sphere. Recent works on view synthesis [136, 3, 76] show that it is beneficial to represent the appearance in 3D

scene with high-dimensional deep features instead of simple RGB colors. Representing higher dimensional features at each MSI sphere is memory intensive as representing f-dimensional features at each point in each MSI sphere leads to a memory complexity of $\mathcal{O}(m \times d \times f)$ which is not feasible even for moderate values of $f > 10$. In this chapter,

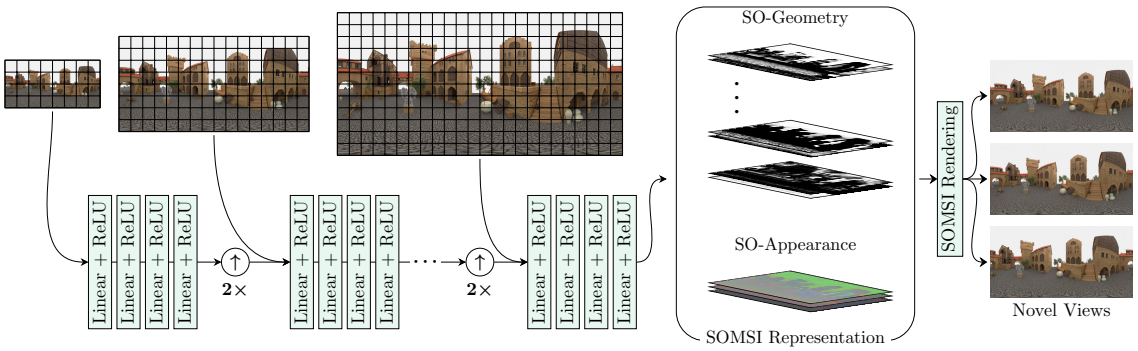


Figure 4.2: **Approach Overview.** We learn an MLP that takes spherical locations and colors from multi-resolution reference view images; and estimates SOMSI scene representation at those locations. We can then efficiently render novel view images from the learned SOMSI.

we propose a novel MSI representation called *Soft Occlusion Multisphere Images* (SOMSI) which scales much better with increasing appearance feature dimensionality. The key to our technique is decoupling appearance features from scene geometry in MSI representation. In SOMSI, we use standard MSI ERPs to represent geometry and soft occlusions; and use a small set of occlusion layers/ERPs to represent appearance features. Our key insight is that much of the points in MSI spheres are empty and we can represent multisphere appearance features with a much smaller set of occlusion appearance features.

SOMSI Representation

As illustrated in Figure 4.3, SOMSI representation has two sets of ERPs: *SO-Geometry* and *SO-Appearance*. SO-Appearance consists of soft occlusion appearance features $S \in \mathbb{R}^{m \times k \times f}$, where f denotes the size of color/appearance descriptors and k the number of occlusion layers. Each of the k ERPs denotes the scene appearance at a specific occlusion layer. The first ERP represents all the visible surface appearances in the reference view; second ERP represents the appearance of occluded surfaces that are behind the visible surfaces and the third ERP represents the further occluded surfaces and so on. In practice, we observe that 3 layers ($k = 3$) are enough to represent most occluded content in general scenes. In short, SO-Appearance represents the scene appearance features with much smaller set ($k = 3$) of occlusion layers in comparison to larger number ($d > 32$) of ERPs in a standard MSI representation.

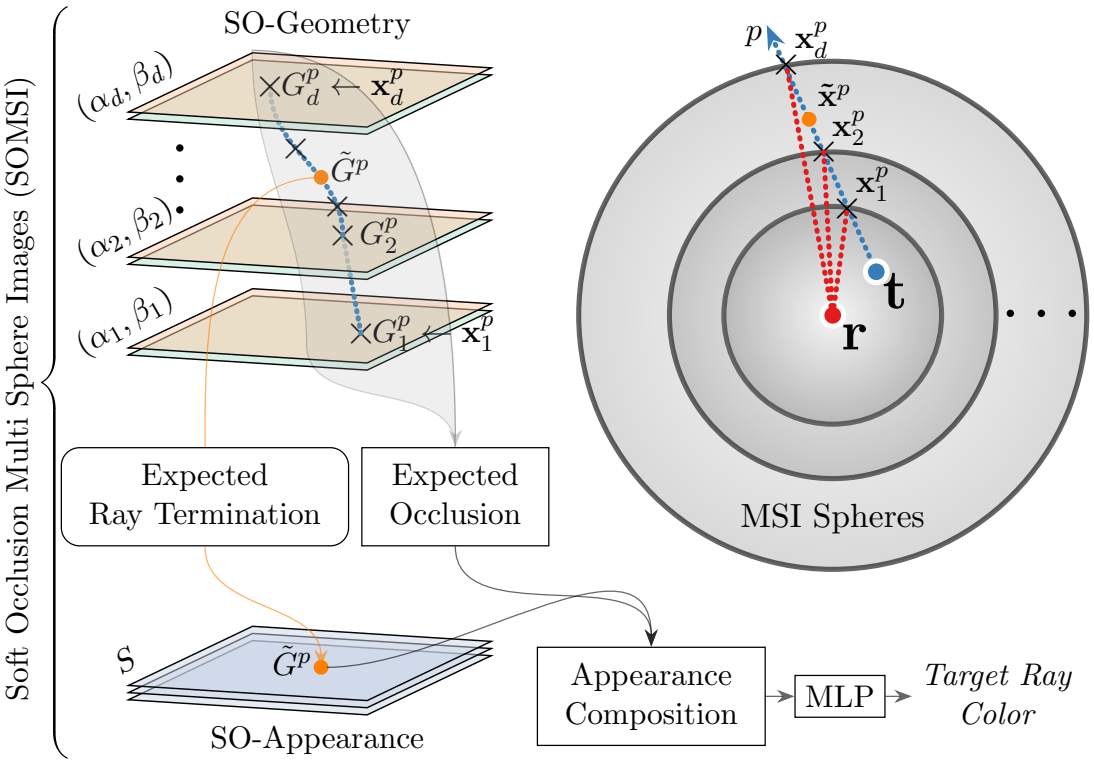


Figure 4.3: **SOMSI Rendering.** Illustration of SO-Geometry and SO-Appearance planes in our SOMSI scene representation in the reference view at **r**. Rendering a target ray p involves computing expected target ray termination and expection occlusion level, using which we can composite the appearance features to estimate the final target ray color.

SO-Geometry, on other hand, represents scene geometry using all the d spheres. Specifically, SO-Geometry consists of multisphere transparencies $\alpha \in \mathbb{R}^{m \times d}$ and soft occlusion masks $\beta \in \mathbb{R}^{m \times d \times k}$. As the name indicates, transparencies α represent the sphere transparencies like in standard MSI representation. The k-dimensional soft-occlusion mask β_i^p of a 3D point at i^{th} sphere along ray/pixel [2] p represents the occlusion level of that 3D point $\mathbf{x}_i^p$ in a soft manner. For example, β_i^p $[0, 1, 0]$ denotes that $\mathbf{x}_i^p$ belongs to a second occlusion level i.e., occluded by one visible surface. The soft occlusion masks β provide the association of planes in SO-Geometry and SO-Appearance. SO-Geometry and SO-Appearance planes have the memory complexity of $\mathcal{O}(m \times d \times k)$ and $\mathcal{O}(m \times k \times f)$ respectively. Since $k << d$, the combined memory complexity of these two will be smaller than representing f-dimensional features at each of the d planes, which has $\mathcal{O}(m \times d \times f)$ memory complexity. This makes SOMSI representation scale better with higher dimensional features

[2]Note: We often refer to pixels as rays since ERP pixels correspond to a specific ray direction.

while retaining the fast rendering times of MSI representation.

SOMSI Rendering

Novel view rendering with SOMSI is not a simple alpha composition process, (Eqn. 4.1) as we do not represent appearance features at each of the MSI planes. One can simply convert SO-Appearance features into full MSI planes, and then can use alpha composition for novel view rendering. Converting SO-Appearance features into full MSI feature planes would again incur $\mathcal{O}(m \times d \times f)$ memory complexity which we want to avoid. In contrast, we propose a technique that directly utilizes SO-Appearance features for rendering. Our SOMSI rendering along each ray from the target center has three main steps:

1. Computing expected ray termination on the target ray
2. Expected occlusion along the target ray, and
3. Using estimated ray termination and occlusion, to composite novel view appearance features.

from SO-Appearance layers. Let us discuss each of these steps in detail.

1. Expected Ray Termination. Following the same notation as in Sec. 4.3 and as illustrated in Figure 4.3, let us suppose we learn an SOMSI centered at reference $\mathbf{r}$ and we want to do the rendering from target location $\mathbf{t}$. Consider the rendering of target appearance features along the sample target ray direction p, as shown in Figure 4.3. Similar to standard MSI rendering, we first compute these target ray intersections $\mathbf{x}_i^p \in \mathbb{R}^2; i \in \{1, ..., d\}$ with the d MSI spheres (in reference view) using standard sphere-ray intersection technique [91]. Each of these points $\mathbf{x}_i^p$ has the corresponding 2D points in SO-Geometry ERP planes with locations $G_i^p \in \mathbb{R}^2$, opacity α_i^p and soft occlusion β_i^p as shown in Figure 4.3. These points $\{G_i^p\}_{i=1}^d$ lie on an epipolar line in the reference view. We compute the 2D ERP location (of ray p in the reference camera) $\tilde{G}^p \in \mathbb{R}^2$ of the target ray p as:

$$\tilde{G}^p = \sum_{i=1}^{d} \alpha_i^p \, G_i^p \prod_{j<i} (1 - \alpha_j^p), \tag{4.2}$$

which is an alpha composition of 2D ERP locations $\{G_i^p\}_{i=1}^d$ instead of color values as in standard MSI rendering (Eqn. 4.1). The expected ray on the Epipolar line, $\tilde{G}^p$ is a 2D location in SOMSI. This point effectively computes the location of target ray termination in the reference camera (depicted as orange dots in Figure 4.3). Note that, one can equivalently compute the ray termination in $3D$ by alpha compositing the ray-sphere intersection points and project the result to the reference ERP to get $\tilde{G}^p$.

2. Expected Occlusion. The expected ray termination $\tilde{G}^p$ at the reference view provides the correspondence, in the reference view, for the target ray p. Thus, we can use this ERP location $\tilde{G}^p$ to get the corresponding appearance features from SO-Appearance. Since SO-Appearance has different occlusion layers, we also need to estimate expected

occlusion level for the target ray p. We compute the expected occlusion level $\tilde{S}^p \in \mathbb{R}^k$ using the same alpha compositing formulation, but compositing the soft-occlusion masks β in back-to-front manner:

$$\tilde{\beta}^p = \sum_{i=1}^{d} \alpha_i^p \, \beta_i^p \prod_{j<i} (1 - \alpha_j^p). \tag{4.3}$$

3. Appearance Composition. To compute an appearance feature for the target ray/pixel p, we first bilinearly interpolate the SO-Appearance features S at the nearest grid points resulting in a multi-layer appearance feature $\tilde{S}^p \in \mathbb{R}^{k \times f}$. We then multiply this feature map with the estimated soft occlusion mask $\tilde{\beta}^p$ to obtain a single appearance feature for the target ray: $\tilde{Q}^p = \tilde{S}^p \tilde{\beta}^p$. Multiplying $\tilde{\beta}^p$ with $\tilde{S}^p$ effectively chooses the SO-Appearance feature $\tilde{Q}^p \in \mathbb{R}^f$ at the expected occlusion level in a soft manner. In summary, we render appearance features along a target ray by first computing the corresponding ERP location in the reference view and then choosing the appearance feature at that ERP location and the expected occlusion level.

Synthesizing Novel View ERPs. We still need to convert the appearance feature $\tilde{Q}^p$ into RGB color for the target ray p. We use a simple MLP to convert $\tilde{Q}^p$ into target ray color $\tilde{C}^p \in \mathbb{R}^3$. We repeat the same SOMSI rendering processing for all the target rays followed by conversion to color values to obtain the complete novel view spherical image.

4.4.2 Modeling view dependent effects.

Color based MSI approaches such as MODS [4] can not robustly model view dependent effects such as specular reflections in target views as color values are baked in the reference MSI. Following the neural basis decomposition approach [137], our SO-Appearance model can also incorporate view dependent effects in a natural way. A neural basis decomposition approach replaces each RGB color value (in the MPI or MSI) by reflectance coefficients ($\omega \in \mathbb{R}^{e \times 3}$, where e is the number of reflection coefficients). The coefficient images can be combined into a single color image using learned basis functions $\{\gamma_i : \mathbb{R}^3 \mapsto \mathbb{R}^1\}_{i=1}^{e-1}$, as follows $C = \omega_0 + \sum_{i=1}^{e-1} \gamma_i \, \omega_i$. γ_i are the scalar outputs of an MLP network that takes viewing direction as input.

Applying this view dependent modelling scheme directly in a standard MSI representation leads to a high memory complexity of $\mathcal{O}(m \times d \times 3 \times e)$ as we need to represent e-dimensional reflectance coefficients for each color value at each MSI point. For MPIs, to overcome this issue, NeXMPI [137] proposes a *coefficient sharing* strategy by grouping different MPI planes and using same reflectance coefficients within each MPI plane group. Even though one could use similar grouping strategy in MSI representation, the grouping of MSI spheres is not guided by the underlying geometry. Our SO-Appearance layers provides a more physically meaningful grouping of MSI spheres based on occlusions and thus provide a natural choice to efficiently represent reflectance coefficients with $\mathcal{O}(m \times k \times 3 \times e)$ instead of $\mathcal{O}(m \times d \times 3 \times e)$. More specifically, just like appearance

features, we need to represent reflectance coefficients only at k occlusion layers in SO-Appearance instead of much larger d spheres in MSI. In the experiments, we observe that using modelling view dependent effects with reflectance coefficients in SO-Appearance improves the view synthesis results (Table 4.4).

4.4.3 Learning SOMSI with Coordinate MLP

We learn a scene-specific SOMSI from a given set of input spherical images captured at different 3D locations in a scene. Following the recent success of coordinate MLPs for view synthesis [83, 80], we make use of MLPs that takes 2D ERP pixel locations as input and produces SOMSI scene representation at those locations w.r.t. a reference view. As illustrated in Figure 4.2, MLP takes the spherical coordinates $[\theta, \phi]$ for a point r along with reference image color I^r_{ref} at that location as inputs, and produces SOMSI representation at that location:

$$\mathcal{N} : (\zeta(\theta, \phi), I^r_{ref}) \mapsto (\alpha^r, \beta^r, S^r), \tag{4.4}$$

where $\zeta(\theta, \phi)$ denotes the Fourier embedding of the coordinates as used in recent coordinate networks such as [83]. Contrasting to existing works that only feed Fourier embedded coordinates into MLPs, we also input RGB color values I_{ref} as well to the MLP. We observe faster training convergence and better Novel View Synthesis quality with additional I_{ref} as input to the network. Another distinction in our network is the use of multi-resolution coordinate maps. As shown in Figure 4.2, we start with low-resolution grid as network input and upsample the grid locations as well as input spherical image after every few linear+ReLu layers. We again observe that this multi-resolution strategy resulted in better results compared to using a single resolution grid. Once we learn an SOMSI for a given scene, we can dynamically render spherical images from arbitrary novel view locations using the SOMSI rendering technique described in the previous section.

Training

We train the network $\mathcal{N}$ on a specific scene by randomly sampling a mini-batch of input cameras and rendering their corresponding held-out novel views. Our rendering pipeline is fully differentiable (due to our soft occlusion formulation) and thus allows for training with back-propagation. The network parameters are learned by minimizing the L_2 distance between the predicted novel view $\hat{I}^t$ and its respective groundtruth ERP image I^t for every camera in the mini-batch.

4.5 Experiments

We analyzed our SOMSI technique for Spherical Novel View Synthesis on both synthetic and real-world datasets; with both structured (input views captured with cameras placed on a regular grid) and unstructured spherical light fields.

Figure 4.4: **Sample visual results.** Novel views on the *Sea Port* (row-1), *Coffee Area* (row-2) and *Residential* (row-3) datasets. Coffee Area is a real dataset captured with Ricoh-Theta-S camera, while the rest are synthetic. In the first row, MODS produces blurred images and struggles with thin structures. S-NeRF creates images with even less details than MODS (see the cobblestones) and far away objects (eg. the tree beyond the gate) are not well reconstructed. Specular reflections (2nd row) and thin structures (3rd row) are better captured with our method. For a better visualization, we encourage the reader to zoom-in on the insets.

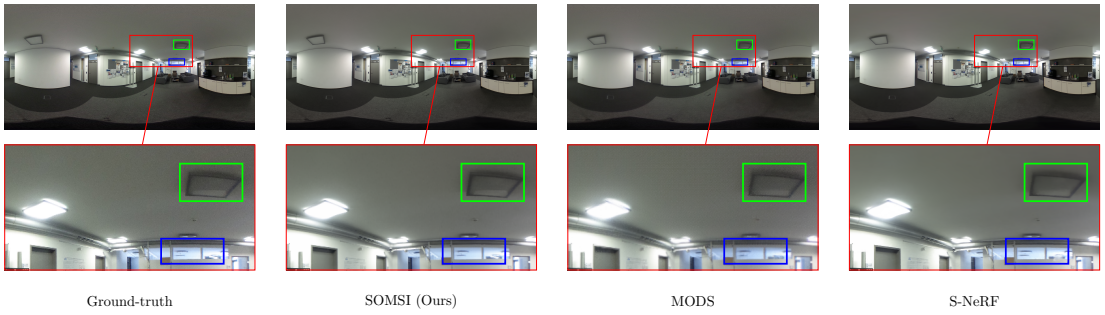


Figure 4.5: **Sample visual results, Coffee Area:** *First row:* Ground-truth and novel views generated by SOMSI, MODS and S-NeRF, respectively. *Second row:* Close-ups on the ceiling. Specular reflections are better captured with our method. For a better visualization, we encourage the reader to zoom-in on the insets.

Baselines

We compare SOMSI with two baseline techniques: MODS [4] and S-NeRF [83]. MODS [3] is a spherical view synthesis method that renders novel views from an Omnidirectional Stereo (ODS) input pair. We adapted MODS to work with a pair of ERPs as input, like

[3] A brief introduction the MODS approach is presented in Chapter 2.

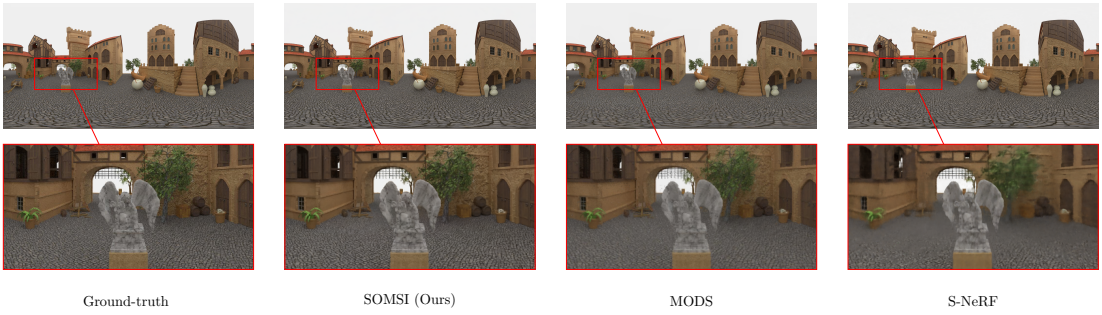


Figure 4.6: **Sample visual results, Sea Port:** *First row:* Ground-truth and novel views generated by SOMSI, MODS and S-NeRF, respectively. *Second row:* Close-ups on the statue. MODS produces blurred images and struggles with thin structures. S-NeRF creates images with even less details than MODS (see the cobblestones) and far away objects (eg. the tree beyond the gate) are not well reconstructed.

Figure 4.7: **Sample visual results Coffee Area:** *First row:* Ground-truth and novel views generated by SOMSI, MODS and S-NeRF, respectively. *Second row:* Close-ups on the ceiling. Specular reflections are better captured with our method. For a better visualization, we encourage the reader to zoom-in on the insets.

in our setting. Given a spherical light field, we train MODS by randomly sampling a triplet of spherical images where the first two serve as the input and the last one is used as target. During test, we render target novel views by using the first and last training cameras as input. We omitted the inverse-transform regularization in MODS [4] baseline as the MODS work itself noted that this adversely effects the view synthesis results.

Our second baseline is a NeRF [83] based technique. The original NeRF technique is designed to work with perspective images. We trained NeRF with spherical images by changing the camera model from pin-hole (perspective) to spherical. We refer to this spherical NeRF baseline as 'S-NeRF' in the experiments. We train all three techniques (MODS, S-NeRF and SOMSI (Ours)) on a per-scene basis.

	SOMSI (Ours) 13 ms		MODS [4] 10 ms		S-NeRF [83] 2710 ms	
Scene	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑
Residential	37.01	0.946	33.37	0.907	36.29	0.944
Replica	39.54	0.986	35.91	0.970	42.51	0.992
Coffee Area 1,2	32.48	0.872	28.73	0.794	32.59	0.874
Sea Port	27.67	0.825	23.79	0.626	27.76	0.825

Table 4.1: **Comparison with state-of-the-art.** SOMSI (Ours) can produce considerably higher quality results compared to MODS [4] with similar runtime. SOMSI produces on par results with S-NeRF [83] while being 2 orders of magnitude faster.

Datasets

We experiment with a total of 20 scenes from the following 4 datasets:

- *Replica* [120] public dataset contains 18 different photorealistic reconstructions of indoor scenes. We used 12 randomly selected scenes. We used the HabitatAPI [106] to render a 5×5 grid of spherical images, where cameras are placed $20\ cm$ apart.
- *Residential* dataset consists of 3 synthetic scenes depicting residential houses that were created using Unity3D [44]. We create 3×3 views per scene with cameras that are $20\ cm$.
- *Coffee Area* dataset consists of 4 real-world scenes that are self-captured using Ricoh-Theta-S spherical camera. The first 2 of scenes have mainly diffuse objects, while the last 2 contain reflective objects. We use COLMAP [108] to calibrate the cameras. See details in the supplementary material.
- *Sea Port* dataset is an synthetic 3D model of a medieval sea port. This dataset is created by improving assets that where borrowed from blendwap. This dataset consists of a 5×5 spherical light field with $20cm$ camera baseline. axes.

Training and evaluation

Following MODS [4], we use 320×640 resolution spherical images for training and evaluation. Training our model takes about 18 hours on RTX-2080Ti GPU. SOMSI can be trained to up to 1024×2048 resolution images on a single RTX-A100 GPU. See the supplementary material for more details. For quantitative evaluation, we compare the predicted novel views with the held-out ground-truth views and then compute the commonly used metrics in view synthesis literature: Peak Signal-to-Noise Ratio (PSNR)

and Structured Similarity Index (SSIM) [135]. Unless specified otherwise, we train our main model with: number of spheres $d = 64$, occlusion layers $k = 3$ and the appearance features size $f = 24$.

Comparisons

Table 4.1 shows the comparison of different metrics across different datasets and techniques. As the PSNR and SSIM metrics clearly demonstrate, SOMSI has considerably and consistently better PSNR and SSIM across all the datasets when compared to MODS [4]. SOMSI has on par metrics compared to the spherical version of NeRF (S-NeRF) [83]. S-NeRF is a strong baseline that stores the scene using MLPs in an implicit manner. As a result, the rendering speed is quite slow (2710ms) compared to SOMSI (13ms). Both SOMSI and MODS learn explicit scene representation enabling fast rendering. In summary, SOMSI provides a good balance between rendering speed and view synthesis quality. That is, SOMSI can produce on par view synthesis results with S-NeRF while being two orders of magnitude faster (similar rendering time as MODS).

Visual Results

Figure 4.6 depicts the ground-truth image and the novel views generated by SOMSI, MODS [4] and S-NeRF [83] for the synthetic Sea Port dataset. MODS produces images that are overall blurred and struggles with thin structures, as can be seen on the bars of the gate behind the statue. Additionally, artifacts are visible around the wing of the statue. S-NeRF manages to capture the bars of the gate and artifacts are not visible. However, in general it delivers images with even less details than MODS. This becomes clear as the cobblestones are blended together and object boundaries are difficult to distinguish. Moreover, S-NeRF fails to reproduce the tree beyond the gate, where distortions are easily noticeable. SOMSI, on the other hand, successfully preserves details of all structures independent of how far they are from the camera.

Figure 4.5 shows one of the manually captured spherical light fields named Coffee Area. Here, all three approaches deliver similar images at a first glance. Nevertheless, as can be seen in the second row of Figure 4.5 our method outperforms both MODS and S-NeRF in terms of scene fidelity. More specifically:

- SOMSI successfully captures specular reflections as shown in the green boxes. One can easily distinguish the edge of the reflection in the image rendered by SOMSI whereas in the images produced by MODS and S-NeRF the reflection smoothly fades out.

- Our approach is again better at reproducing small details on surfaces far from the camera, as shown in the blue boxes. While MODS lacks details and S-NeRF creates distortions (see lower right corner of the box), SOMSI faithfully reproduces the scene.

Finally, Figure 4.7 shows an example of the Residential dataset. Here again, MODS delivers images that are overall blurred and therefore lack detail (see upper right corner on the second row of Figure 4.7 for the output of MODS). As for the results on the two datasets discussed above, S-NeRF produces images with distortions in some areas. This is clearly visible on the zoomed-in window. In contrast, our method produces an image that is neither blurred nor distorted.

Ablations

We perform ablations with the appearance feature dimensions f and the number of occlusion layers k in SO-Appearance features. Table 4.2 shows metrics with different feature dimensions $f = 3, 12, 24$ on three scenes from Replica, Residential and Sea Port. Results clearly show that both PSNR and SSIM improves with increasing feature dimensions. We observe minimal performance improvements with even higher feature dimensions $f > 24$. This observation is inline with recent view synthesis studies [3, 136] on perspective images. These results demonstrate the use of high-dimensional features to model appearance details in an MSI instead of commonly used RGB values thereby justifying the need for our efficient SOMSI representation. Using SOMSI representation makes it feasible to increase the appearance feature dimensions without incurring significant memory or runtime costs, compared to representing appearance features using standard MSI representation.
Vanilla RGBA MSI baseline without MLP. We have experimented with a baseline that learns an MSI scene representation, directly without employing MLPs. In this setting we initialize and directly optimize an $m \times d \times 4$ dimensional MSI via gradient descent. Although cheaper to predict, it fails to produce realistic MSIs, as shown in Figure 4.8. PSNR score for this baseline is much lower than our method (by at least 8 PSNR drop on Sea-Port and Residential datasets). Therefore, we conclude, the inductive bias of the MLP is crucial in learning realistic MSIs. Due to its poor performance, we do not treat this approach as a baseline and, instead we use more robust baselines such as MatryODShka [4] and Spherical NeRF [83].

Table 4.3 shows the view synthesis metrics with the different number of occlusion levels k. We notice some increase in performance with using 3 levels compared to 2 levels. Using more occlusion levels did not show any considerable improvement in performance. This shows that 3 occlusions levels are usually enough to represent a scene in SOMSI representation.

View Dependent Effects

For the two scenes in Coffee Area that has specular objects, we experiment with learning reflectance coefficients in SO-Appearance as described at the end of Section 4.4.1. Table 4.4 shows the PSNR and SSIM metrics on the two Coffe Area scenes with different number of reflectance coefficients. Results clearly show that the view synthesis quality improves with more reflectance coefficients. Even though this is not a surprising result, this further

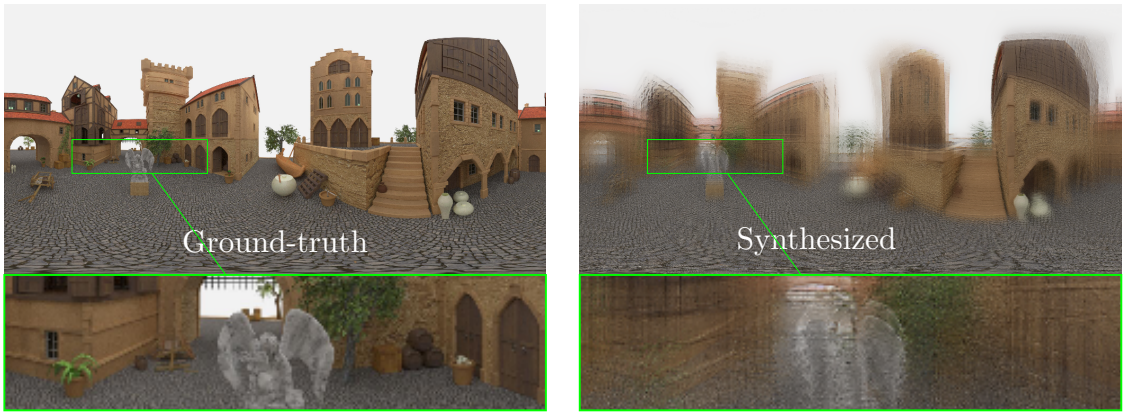


Figure 4.8: **Sample Vanilla-MSI Result.** Novel view (right) generated using Vanilla-MSI is quite blurry compared to GT (left). Producing realistic MSIs without the inductive bias of MLPs is challenging. Please zoom in for details.

	f=3		f=12		f=24	
Scene	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑
Residential	36.96	0.943	36.82	0.944	**37.01**	**0.946**
Replica	38.75	0.980	38.97	0.981	**40.23**	**0.986**
CoffeeArea 1,2	32.33	**0.873**	32.44	**0.873**	**32.48**	0.872
Sea-Port	27.27	0.802	27.55	0.822	**27.67**	**0.825**

Table 4.2: **Ablation of appearance feature dimensionality f.** Metrics show that higher feature dimensions results in better performance thereby justifying the need for our SOMSI technique that can effectively represent higher-dimensional appearance features with occlusion layers instead of a full set of MSI layers.

	$k=2$		$k=3$		$k=5$	
Scene	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑
Sea-Port	27.30	0.809	27.32	0.813	27.44	0.814
Replica	36.51	0.977	37.13	0.979	36.78	0.981

Table 4.3: **Ablation of occlusion levels k.** Metrics show that 3 occlusion levels are good enough to represent the scene and to obtain high-quality view synthesis results. Due to limited compute resources, here we used only the first 3 scenes from Replica.

	$e = 1$		$e = 4$		$e = 6$	
Scene	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑	PSNR ↑	SSIM ↑
CoffeeArea-3	33.66	0.891	33.70	0.892	**35.86**	**0.902**
CoffeeArea-4	30.49	0.865	30.53	0.867	**30.61**	**0.869**

Table 4.4: **The effect of number of reflectance coefficients e for view dependent effects.** Metrics demonstrate the use of higher number of reflectance coefficients further emphasizing the need for modelling higher dimensional features in MSI representation.

demonstrates the use of learning higher dimensional features in MSI (either appearance features or reflectance coefficients or both) compared to only RGB color values.

Limitations

One of the limitations of our technique is that the scene representation network is optimized independently for different scenes. This assumes a sufficient number of training views for each scene which may not be available for some scenes in practice. A more practical approach would be to learn a network prior that works across different scenes. A main challenge in learning such as dataset prior is that there exists no large scale spherical image dataset with diverse scenes to learn meaningful priors.

4.6 Network Architecture and Training Details

4.6.1 Network Architecture

We use a multi-scale MLP network that is implemented as 1×1 convolutional network. The network takes the reference views' 2D positional encodings of each pixel along with RGB color values. These inputs are then processed through multiple stages to produce the desired SOMSI scene representation. Note that each pixel (u, v) in the reference Equirectangular Projection (ERP) has a unique viewing direction (θ, ϕ) on the unit sphere associated with it; see preliminary in section 2.

Table 4.5 summarizes the architecture of our network: It is composed of convolutional blocks, each block consisting of four 1×1 convolutions, each followed by ReLU. The positional encodings and color values of the reference ERP are fed to each convolution block at different resolutions. The output of $Block\ i$ is bilinearly upsampled by a factor of 2 and is given as input to $Block\ i + 1$. Please refer to Figure 4.2 for an overview of SOMSI's architecture.

In Table 4.5, $\mathcal{E} = 4 \times l + 2$ is the positional encoding size, for a given number of octaves l. The number of channels in the last block corresponds to the total number of outputs per

	Input channels	Output channels	Input scale	Output scale
Block 1	$\mathcal{E}+3$	$\mathcal{F}[1]$	$H/2^7$	$H/2^6$
Block 2	$\mathcal{E}+3+\mathcal{F}[1]$	$\mathcal{F}[2]$	$H/2^6$	$H/2^4$
Block 3	$\mathcal{E}+3+\mathcal{F}[2]$	$\mathcal{F}[3]$	$H/2^5$	$H/2^3$
Block 4	$\mathcal{E}+3+\mathcal{F}[3]$	$\mathcal{F}[4]$	$H/2^4$	$H/2^2$
Block 5	$\mathcal{E}+3+\mathcal{F}[4]$	$\mathcal{F}[5]$	$H/2^3$	$H/2^1$
Block 6	$\mathcal{E}+3+\mathcal{F}[5]$	$\mathcal{F}[6]$	$H/2^2$	$H/2^1$
Block 7	$\mathcal{E}+3+\mathcal{F}[6]$	$\mathcal{F}[7]$	$H/2^1$	H
Output Block	$\mathcal{F}[7]$	$k\times f+d\times(k+1)$	H	H

Table 4.5: **SOMSI Network architecture.** The input to SOMSI is the concatenation of reference image RGB values and positional encodings of its 2D pixel positions.

pixel, i.e. d opacity values, $d\times k$ visibility/occlusion and $k\times f$ multi-occlusion appearance features. The number of features per channel is decreased linearly from $\mathcal{F}[1]=256$ to $\mathcal{F}[7]=128$.

Unless specified otherwise, we use the following settings for our experiments: number of spheres $d=64$, number of occlusion layers $k=3$ and number of octaves $l=4$.

Training

We trained our model using a batch size $b=2$ for 30000 epochs. We use the Adam [69] optimizer with the following settings: $lr=0.001$, $\beta_1=0.9$, $\beta_2=0.999$. The network is implemented in the PyTorch [89] Deep Learning framework. The network can be trained in about 18 to 24 hours on an NVIDIA GPU that has more than 15GB memory.

4.7 Calibrating Spherical Light Fields through Perspective Cut-outs

The Structure-from-Motion (SfM) toolbox from Schönberger *et al*. [108] (known as COLMAP) has become the standard tool for calibrating multi-view datasets. COLMAP works for pinhole and fisheye cameras. We calibrate our spherical datasets (*Coffee Area 1-4*) by first extracting virtual perspective views from each spherical image and applying COLMAP on the collection of perspective images. We recover the intrinsic and extrinsic parameters of the virtual perspective cameras using COLMAP's SfM module.

The next step is to determine the pose of all spherical cameras based on their virtual perspective counterparts. This is possible because the output camera poses of COLMAP and their corresponding spherical cameras are related by a rotation matrix R_s^p. For a perspective image whose principal axis intercepts the unit sphere at coordinates (θ,ϕ), R_s^p

can computed as: $R_s^p = R_x \times R_z$, where R_x is a rotation around the x axis by angle ϕ and R_z is a rotation around the z axis by $\frac{\pi}{2} - \theta$.

Perspective to sphere rotation matrices $R_p^s = (R_s^p)^T$ are used to calculate the rotation component of the spherical camera poses: $R_w^s = R_p^s \times R_w^p$. Translation vectors are adapted accordingly: $\mathbf{t}_w^s = R_p^s \times \mathbf{t}_w^p$, where $\mathbf{t}_w^s$ is expressed in the spherical camera coordinates system.

4.8 Conclusion

In this chapter, we propose a novel multisphere representation called SOMSI that can effectively model high-dimensional appearance features in MSI representation that are commonly used for Spherical Novel View Synthesis. The key is to represent features in occlusion layers instead of full set of MSI spheres. We presented a novel SOMSI rendering scheme that retains the fast rendering of standard MSI representation while producing high quality view synthesis. SOMSI also produces on par results with NeRF technique, while being 2 orders of magnitude faster.

5 Generative View Synthesis: from Single-view Semantics to Multi-view Images

This chapter introduces *Generative View Synthesis* (GVS), a novel technique that can synthesize multiple photorealistic views of a scene given a single 2D semantic map. Unlike the methods introduced in Chapters 3 and 4, GVS requires minimal input and grants users more control over the properties of the 3D scene. The underlying 3D scene can be edited by simply modifying the input 2D semantic map. In addition to the input semantic map, GVS takes a *style image*, that determines the global scene characteristics.

The work presented in this chapter has been published in the following article:

1. Tewodros Amberbir **Habtegebrial**, Varun Jampani, Orazio Gallo and Didier Stricker Generative view synthesis: From single-view semantics to novel view images. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Project page `https://gvsnet.github.io`.

Contents

5.1 Introduction **80**
5.2 Related Works **82**
5.3 Generative View Synthesis Model **83**
5.4 Dataset, Experiments and Evaluations **87**
5.5 Conclusion **95**
5.6 Additional Experimental Details **95**

5.1 Introduction

In previous chapters we have developed techniques that can reconstruct novel views of a scene, given a set of input images. The input images and their camera poses serve as a *specification* for the desired scene. Those NVS approaches aim to faithfully generate the scene as specified by the input camera images and poses. However, these approaches offer the user no control over the scene being generated. Moreover, these techniques require additional steps such as capturing images and performing camera calibration. In this chapter, we aim to solve an Image-based Rendering (IBR) problem which could give users more creative freedom and requires minimal input. To this end, we introduce and solve the *Generative View Synthesis* problem, where a 2D semantic map is converted into 3D-consistent multi-view images. In the GVS pipeline, the user can alter the scene geometry by editing the input semantic map. The proposed GVS approach takes a *style image*, with which the user can dictate the global appearance/style properties of the scene.

The rising demand for digital content, together with the widespread availability of high-quality digital cameras, has fueled the need for tools and algorithms to democratize content creation. A prominent example of one such technology is Novel View Synthesis (NVS), which allows the artist to render a scene from new viewpoints, using as few as a single image [24, 136]. Photorealistic images can also be generated by editing a simplified representation of the scene, such as a semantic map, followed by image-to-image translation [88], but the viewpoint cannot be manipulated. Our GVS approach combines the advantages of both worlds. Given a single semantic map, which is easy to edit and requires no image capture, GVS can generate RGB images of the same layout, but from new, arbitrary viewpoints. Not surprisingly, GVS also inherits the challenges of both: generating RGB values from a bare semantic map is an ill-posed problem that is further complicated by the need for the different output views to be photometrically and geometrically consistent.

One could tackle this problem with the sequential application of existing techniques. That is, we can first convert the single-view semantic map into an RGB image using image-to-image translation techniques [62], and generate novel RGB views using monocular Novel View Synthesis techniques [136]. However, we observe that this may fail at preserving the scene's structure accurately, as shown in the animation in Figure 5.1.

Our key insight is that semantic maps are particularly informative about the structure of a scene, despite offering no information about its photometric properties. Semantic segments, in fact, carry more unambiguous information about occlusion boundaries. This is in contrast with RGB images, where edges can also result from texture. We leverage this observation to preserve geometric consistency between multiple output views. Specifically, instead of converting the semantic map to an RGB image, we propose to first uplift the 2D semantics into layered 3D semantics with a structure similar to Multiplane Images (MPI) for RGB images [121, 149]. We call this structure *lifted semantics*. Unlike MPIs, to relax the memory requirements and for translation efficiency, our lifted semantics use a hybrid representation with a small set of semantic layers and a larger set of opacity layers. We

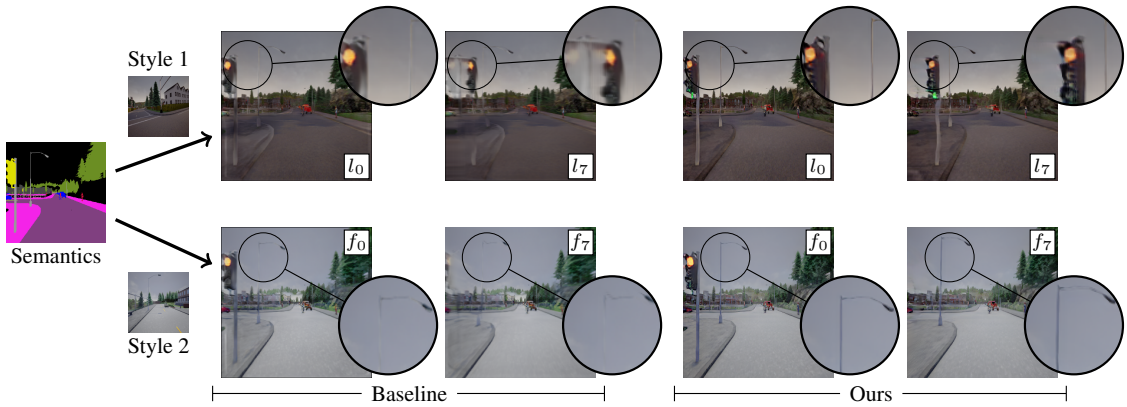


Figure 5.1: **Generative View Synthesis** is a method to generate photorealistic images from novel viewpoints, given just a semantic map and a style image. Here we show lateral (l_{0-7}) and forward (f_{0-7}) camera motion. Because no methods exist to solve this problem, we propose to use SPADE [88] followed by single-image MPI rendering [149] as a baseline. Our method better preserves thin structures and produces geometrically consistent views. *Animated figure. Please view in Adobe Reader and click on the image to see the animation. Other PDF viewers may have issues, in which case please refer to the project page* `https://gvsnet.github.io`.

convert the lifted semantics to layered features, which we refer to as *layered appearance*, and combine them with the opacity layers. Finally, we project the resulting appearance features onto the target views and convert them to RGB images with a small network. The late fusion of the lifted semantics is key to the quality of our results. In summary, our work benefits from, and leverages three main observations:

1. Semantic maps naturally disambiguate between texture edges and geometric discontinuities, thus offering a representation well-suited to generate novel views.

2. As a result, preserving semantic information for as long as possible in the pipeline, rather than converting to RGB early, encourages geometric consistency between the output views.

3. However, projecting the semantic information directly in the output views and converting to RGB causes photometric inconsistencies, which can be addressed by converting the semantics to an MPI-based representation of appearance first.

We perform extensive experimental analysis on three different multi-view datasets: CARLA [30], Cityscapes [28], and Virtual-KITTI-2 [13]. We show both qualitatively and quantitatively that our approach, which compares favorably with strong baseline techniques, produces novel view images that are geometrically and semantically consistent. In addition, we also demonstrate that we can estimate high-quality depth information from single-view semantics.

5.2 Related Works

Novel View Synthesis (NVS) has a rich history that predates the deep learning era [22, 152]. Traditional methods tackle the challenge of generating pixels for unseen viewpoints with proxy geometry [12, 29], or with a significant number of input images [17, 16]. Thanks to learned priors, impressive results with as little as two input images and no additional information are also possible [26, 149, 119, 35]. A key regularization technique at the core of many of these methods is a scene representation consisting of a set of fronto-parallel layers that can be merged down into the target view, after appropriate warping. This approach, which relates to representations proposed by earlier methods [121], is dubbed Multiplane Images (MPI) [149] and is also central to the success of our method, albeit with significant modifications. Recent NVS works go even further and show single-image NVS by learning to predict single-image depth [136, 24, 129], voxel grids [86], or by predicting MPI from single image [128].

While powerful, all these NVS methods require an input RGB image. To better leverage their creative agency, users can also edit a simplified representation of a scene or object, which can then be converted to an RGB image. For instance, sketches can be turned into photorealistic pictures [23, 62]. A scene-level representation that is particularly flexible is afforded by semantic maps. Indeed, a number of works using both traditional tools [64] and deep learning [20, 133, 62] produce impressive semantic-to-RGB results. An example of the engagement this type of technology can enable is offered by the method by Park *et al*. [88], which takes a hand-drawn semantic map and produces an RGB image. Within months of publication, more than 500,000 images were created by web users [1].

Image-to-image translation offers more control over the content generated, but unlike NVS methods, it does not allow to modify the viewpoint. We propose to combine the advantages of the two lines of work. Our method takes a semantic image as input and produces photorealistic images from novel viewpoints. However, compared with the sequential combination of existing methods, our strategy better leverages the information the semantic maps offer: a robust representation of the scene structure. In a concurrent work, Huang *et al*. [58] also propose to leverage semantic information by using it to directly predict disparity, which in turn, is used to build a traditional MPI representation of the scene. Different to that [58], we propose to lift 2D semantics onto a hybrid layered 3D semantics and appearance representation.

5.3 Generative View Synthesis Model

We present *Generative View Synthesis* (GVS), a method that takes a single 2D semantic map as input and generates photorealistic images from novel viewpoints. One way to tackle this problem is by a straightforward combination of existing techniques, that is, 2D semantics to 2D image conversion in the reference view [88], followed by monocular Novel View Synthesis [24, 150, 136] to generate the target views. However, this naïve approach fails at preserving some of the structures observed in the semantic input, as can be observed in Figure 5.1. This is because the semantic map's strong cues about the layout of the scene are lost in the early conversion to RGB. In contrast, GVS carries the semantic information forward, and only converts it to RGB after its projection onto the target viewpoint. This results in photorealistic target views that are geometrically consistent and that better preserve the structures in the input semantics.

Approach Overview. Formally, GVS takes a 2D semantic map $\mathcal{S}^r \in \mathbb{R}^{n\times l}$ in the reference view r, where n is the number of pixels and l the number of labels. It also takes the relative camera pose transformation from source to target view $\theta_{r\to t} \in SE(3)$. The output is a image $I^t \in \mathbb{R}^{n\times 3}$ in the target view t. Additionally, we use an image $Q \in \mathbb{R}^{n\times 3}$ to control the style of the generated images.

We train a network to convert the 2D semantics $\mathcal{S}^r$ to $\hat{\mathcal{S}}^r$, a layered semantics representation that we call *MPI semantics*. MPI semantics are inspired by multiplane images (MPIs) [121, 149], which represent a 3D scene with a stack of 2D layers positioned at m depth levels $(d_1, d_2, ..., d_m)$. We represent the MPI semantics with m layers: $\hat{\mathcal{S}}^r \in \mathbb{R}^{n\times l\times m}$. Each layer contains semantic labels at each pixel, at that layer's depth, and the pixel opacity $\alpha \in \mathbb{R}^{n\times m}$. Multiplane images [149, 121] are a widely used representation for Novel View Synthesis because their 2D nature allows for well-studied and powerful processing techniques such as convolutional neural networks (CNN).

One could project the MPI semantics onto the target views and independently translate each of the target-view semantics into RGB images. This would enforce geometric consistency, but it would not guarantee the consistent appearance of the output images. Therefore, we propose a translation scheme that converts the MPI semantics to layered 3D appearance features. We then project the layered appearance features onto the target views and convert them to the target RGB images. We demonstrate that this approach results in multi-view-consistent images and preserves the input semantic structures. Figure 5.2 (top) shows our approach overview: we uplift the 2D semantics using a *semantics uplifting network* (SUN) and translate the lifted semantics to layered appearance features using a *layered translation network* (LTN). We then project the layered appearance features to novel views and convert projected features using an *appearance decoder network* (ADN) to generate target images.

We find that the use of MPI semantics, however, is intractable because of the large memory footprint of the layered translation network (LTN). A faithful approximation of any content in 3D space, in fact, requires $m > 32$ MPI layers, making the semantics-to-appearance MPI translation infeasible—recall that we have l labels for each layer, and each

pixel. Therefore, we propose a hybrid layered representation. Specifically, we learn only $k < m$ semantic layers and the full set of m opacity layers, which we then combine by learning an association function. We call our overall network for generative view synthesis *GVSNet*.

5.3.1 Semantics Uplifting Network

We first uplift the input 2D semantics to layered 3D semantics using a 2D CNN, which we refer to as *Semantics Uplifting Network* (SUN). As outlined above, instead of converting the semantics to MPI-semantics, we propose to use a hybrid representation.

Hybrid Layered 3D Semantics Representation. Representing semantic information at each of the m depth layers of the MPI has a memory footprint in the order of $O(n \times l \times m)$, where n is the total number of pixels, and l the number of labels. In practice, however, the 3D scene is mostly empty, causing each of the layers in the MPI to be sparse. Therefore, we propose to represent the layered 3D semantics with fewer layers, $k < m$, which we call *lifted semantics*, $\tilde{\mathcal{S}}^r \in \mathbb{R}^{n \times l \times k}$. semantics. For this representation, we use the input 2D semantics as the first layer and predict the remaining layers using the SUN network. In practice, we observe that $k = 3$ layers (including the input semantics) suffice. On the other hand, we do not compress the opacity, α, to fewer layers, as it serves as a proxy to the scene geometry and it only requires a scalar value for each pixel in each layer. That is, we represent opacity α with the original number of MPI layers m. The opacity layers are also shared with the layered 3D appearance features. addition to the lifted semantics.

Because of the mismatch in the number of layers, we need to estimate an association map $\Phi \in \mathbb{R}^{n \times k \times m}$ to convert the lifted semantics $\tilde{\mathcal{S}}^r$ to MPI semantics $\hat{\mathcal{S}}^r$. At each pixel p, we can convert the lifted semantics $\tilde{\mathcal{S}}^r_p \in \mathbb{R}^{l \times k}$ into MPI semantics $\hat{\mathcal{S}}^r_p \in \mathbb{R}^{l \times m}$ representation with the column-normalized association matrix $\Phi^*_p \in \mathbb{R}^{k \times m}$: $\hat{\mathcal{S}}^r_p = \tilde{\mathcal{S}}^r_p \, \Phi^*_p$. Figure 5.2 (bottom-left) illustrates the SUN network that takes 2D semantics in the reference view as input and predicts the lifted semantics $\tilde{\mathcal{S}}^r$, the MPI opacity α, and the association map Φ. Figure 5.3 shows sample lifted semantics layers. We visually observe that the lifted semantics layers roughly correspond to occlusion layers, where the farther layers capture the content occluded by the closer layers.

5.3.2 Layered Translation Network

With the MPI semantics $\hat{\mathcal{S}}^r$, we can render the semantics into the target views $\mathcal{S}^t$, and use any of the recent conditional image generation networks [88] to generate target-view images. We empirically observe that independent translations of semantics to RGB images can result in inconsistent results—even when the structure is consistent, the corresponding texture may vary across views. To remedy this, we estimate layered 3D appearance features and directly translate the lifted semantic layers to appearance feature layers. Differently put, we carry forward for as long as possible a view-independent representation of the scene.

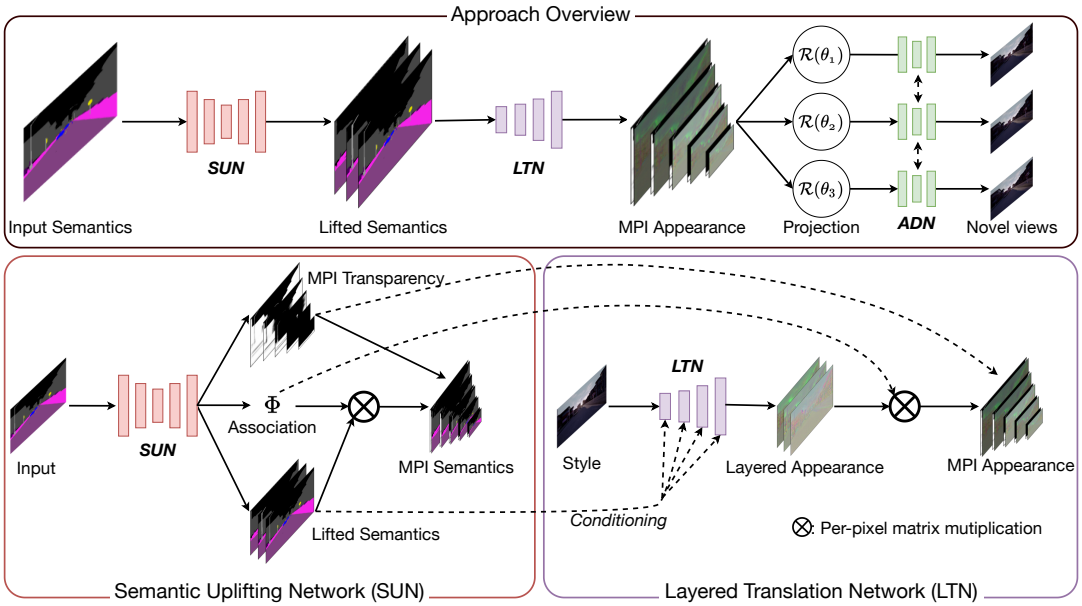


Figure 5.2: **GVSNet Overview.** [Top] Approach overview illustrating the lifting of semantics to layered 3D representation using Semantic Uplifting Network (SUN) and then translating to MPI appearance with Layered Translation Network (LTN). We then project the MPI appearance onto target views and use Appearance Decoder Network (ADN) to generate images. [Bottom] Illustration of different representations and operations in SUN and LTN.

State-of-the-art image-to-image translation networks use large generative adversarial networks (GAN), making it infeasible to translate MPI semantics to MPI appearance features. This is the very reason why we use SUN to predict lifted semantics $\tilde{\mathcal{S}}^r$ instead of MPI semantics $\hat{\mathcal{S}}^r$. We use a *Layered Translation Network* (LTN) to convert k lifted semantics layers $\tilde{\mathcal{S}}^r$ to k-layered appearance features $\tilde{\mathcal{A}}^r \in \mathbb{R}^{n \times f \times k}$, where f denotes the appearance feature dimensionality. LTN is a modified version of SPADE [88]. Like the original SPADE, LTN can also take a style image as input to enable easy manipulation of the appearance of the generated images. However, SPADE takes 2D semantics as input and generates 2D image, while LTN takes multi-layer inputs and produces multi-layer outputs. We then use the same MPI opacity layers α and association map Φ, estimated with SUN, to convert the k-layered appearance features to MPI appearance features with m layers. That is, at each pixel p: $\hat{\mathcal{A}}^r_p = \tilde{\mathcal{A}}^r_p \Phi^*_p$. Figure 5.2 (bottom-right) illustrates the LTN network.

5.3.3 Appearance Decoder Network

We render the MPI appearance features $\hat{\mathcal{A}}^r$ into target-view appearance $A^t \in \mathbb{R}^{n \times f}$ for a given target-view t. We then train a small CNN called *Appearance Decoder Network* (ADN) that converts target-view appearance to the final target image I^t. As an alterna-

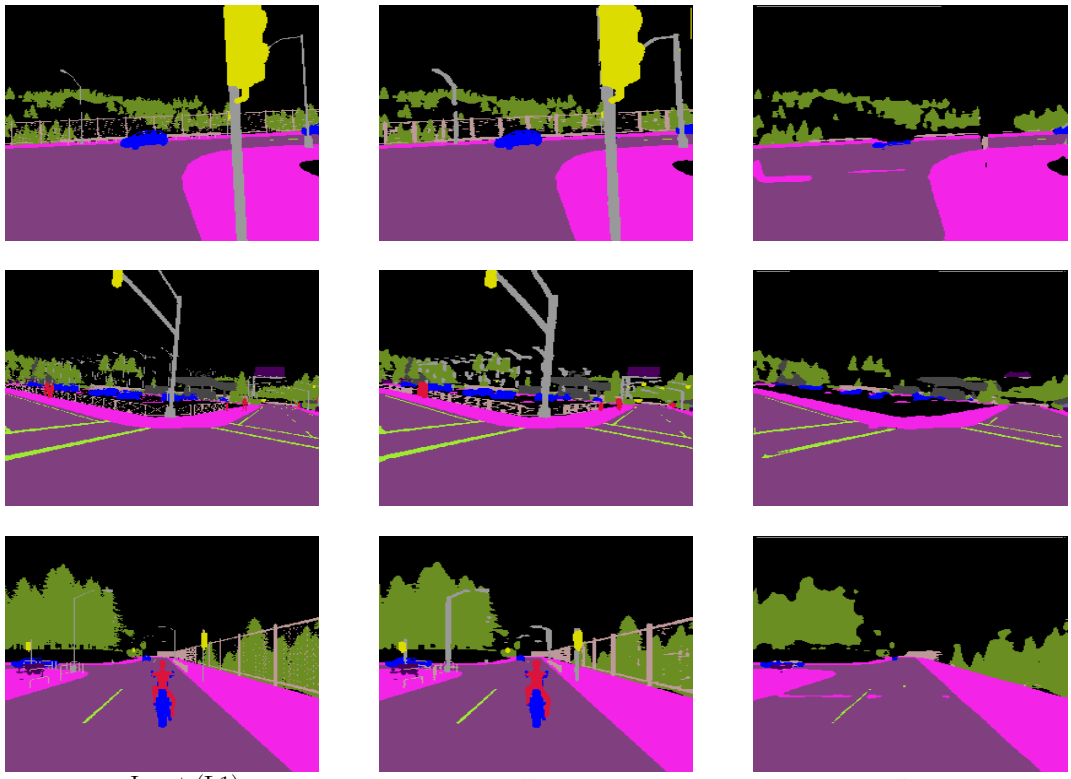


Figure 5.3: **Lifted Semantics.** L1, L2 and L3 correspond to the first, second and third lifted semantic layers respectively. We observe that the lifted semantic layers loosely correspond to occlusion layers, where the later layers capture the content occluded by the earlier layers, see L3 images. Another interesting observation is that, at mid levels (L2), the network *dilates* thin structures (see the poles).

tive, one could directly estimate color MPI using LTN and then just render the color MPI onto target-view to obtain a target image. We empirically observe that projecting high-dimensional (f-dimensional) MPI features can result in better target views in comparison to projecting color MPI. This is because high-dimensional, per-pixel features help mitigate some of the artifacts that arise from the discrete nature of the MPI planes. The high-dimensional features capture neighborhood pixel information and can provide more contextual information to ADN to deal with possible MPI rendering artifacts.

Loss Functions. The overall GVSNet, illustrated in Figure 5.2, has three main subnetworks: Semantics uplifting network (SUN), Layered translation network (LTN) and Appearance decoder network (ADN). To train GVSNet, we use a weighted sum of target segmentation loss $\mathcal{L}_{sem}$, depth loss $\mathcal{L}_{dep}$, target color loss $\mathcal{L}_{col}$ and GAN loss $\mathcal{L}_{gan}$:

$$\mathcal{L} = \lambda_0 \mathcal{L}_{sem} + \lambda_1 \mathcal{L}_{dep} + \lambda_2 \mathcal{L}_{col} + \lambda_3 \mathcal{L}_{gan}. \tag{5.1}$$

$\mathcal{L}_{sem}$ denotes the negative log-likelihood loss for the predicted semantics in the target view. We project the MPI semantics to target-view semantics using standard MPI rendering [149] that involves homography transformation and alpha composition. Refer to [149] for more details on MPI rendering. We then use negative log-likelihood loss $\mathcal{L}_{sem}$ between ground-truth (GT) semantics and predicted semantics in the target views. For the depth loss, we first compute per-pixel inverse depth D^r in the reference view from the predicted MPI opacity α by performing back-to-front alpha composition on the inverse-depth values of each plane at the given pixel. The inverse depth computation from opacity is shown in Equation 5.2. Then the depth loss $\mathcal{L}_{dep}$ is the L1 distance between the GT inverse depth and the computed inverse depth. The target color loss and GAN losses include the same terms as in SPADE [88], that is, perceptual loss on generated color image, discriminator feature reconstruction loss and GAN losses. The main difference to SPADE [88] is that we use these loss functions on generated *target* image with respect to the GT target image.

We train GVSNet in two stages. In the first stage, we pre-train SUN with the target segmentation and depth losses. In the second stage, we train LTN and ADN with the target color loss, while keeping the SUN fixed. We use this two-stage training because the entire network training does not fit on NVIDIA GTX-2080-Ti GPUs, which is what we use for training. However, we could conceptually train GVSNet end-to-end, because all the components are differentiable. The SUN network is a composed of two parts: backbone and three prediction heads (for α, $\tilde{S}^r$, and Φ). The backbone is a UNet [102] style encoder-decoder network with 7 encoding and decoding stages. The prediction heads are convolutional blocks that share their first 3 layers. ADN is a light CNN with 5 encoder-decoder layers. The LTN network is a SPADE [88] network with 7 SPADE-Residual blocks and UpSampling layers. For our experiments, we used $k = 3$ lifted semantics layers, $m = 32$ MPI planes, and $f = 20$ appearance features per pixel. We implemented our model in PyTorch [89] and use the Adam [69] optimizer for training. More details about the training and network architectures are given in the Appendix 5.6.

5.4 Dataset, Experiments and Evaluations

Datasets. GVSNet is fully supervised and thus requires datasets providing two or more views for each scene. It also needs semantic segmentation annotations for at least one view. Given these constraints, we perform experiments on three different datasets: CARLA [30], Virtual-KITTI-2 [13] and Cityscapes [28]. We use a pair of cameras to train GVSNet, treating one of the images as the input view and another one as the target view. In CARLA, we sample pairs from a set of cameras arranged along x-(left-right) and z-(forward) axes. For Virtual-KITTI-2 and Cityscapes, we use stereo pairs with a horizontal baseline.

For CARLA and Virtual-KITTI-2, following the practice in SPADE [88], we use the color image from the input camera as style guidance when generating the target view. In order to make our results comparable to SPADE [88], we do not use style input images in Cityscapes experiments. For the Cityscapes dataset, the ground truth (GT) semantic segmentation is only available for the left camera. We used a pre-trained semantic

Figure 5.4: **Sample Visual Results** showing generated novel view images on CARLA (top), Virtual-KITTI-2 (middle), and Cityscapes (bottom) images.

segmentation network [151] to generate semantics for the right camera images. As there is no ground truth depth in Cityscapes, we computed depth maps by training the DPSNet [60] in a self-supervised manner. In the Cityscapes experiments we use instance masks and the details can be found in the appendix in section 5.6. In all of our experiments we use images at a resolution of 256×256 pixels.

Evaluation Metrics. The results of a GVS system should have three properties: 1. *Semantic Preservation*: The generated image should retain the semantic structures of the input semantics; 2. *Photorealism*: The generated target images should be photorealistic; and 3. *Multiview Consistency*: When rendered across multiple views, the appearance of a given physical object should not change drastically. To measure semantic preservation, we apply a semantic segmentation network (DeeplabV3+ [19]) on the synthesized images and compare its output with the GT semantics in the target view. Specifically, we report mean class accuracy and mean Intersection over Union (IoU) as segmentation metrics. To measure photo-realism, we report the FrÃ©chet Inception Distance (FID) score [51] and Perceptual Distance (PD) [145] metric, which measure the distance between generated target-view images and GT images in VGG [115] feature space. We evaluate multi-view consistency qualitatively and quantitatively by warping two novel views onto a reference camera and comparing the warped images.

Comparisons to Baselines. As this is the first work to tackle GVS, there is no existing baseline technique against which we can directly compare. To properly evaluate our approach, then, we propose sensible baseline methods based on the adaptation and combination of state-of-the-art methods. Specifically, we use a pipeline that converts 2D semantics to an RGB image with SPADE [88] and applies monocular NVS techniques

	CARLA [30]				Virtual-KITTI-2 [13]				Cityscapes [28]	
Method	Cls. Acc. ↑	IoU ↑	PD ↓	FID ↓	Cls. Acc. ↑	IoU ↑	PD ↓	FID ↓	PD ↓	FID ↓
GVSNet (Ours)	**74.34**	**66.43**	**1.74**	**62.06**	**77.13**	**69.62**	**2.08**	**36.21**	**2.76**	**48.72**
SPADE [88] + SM [149]	69.93	60.82	1.95	75.81	74.84	64.71	2.19	41.61	2.82	60.71
SPADE [88]+ CVS [24]	66.84	57.29	1.88	69.24	76.23	67.73	2.12	37.79	2.80	57.46
SPADE [88] + AF [150]	66.15	56.45	1.92	76.89	76.81	68.66	2.15	40.95	2.83	57.15
Target GT Images	77.47	69.67	-	-	83.58	75.39	-	-	-	-

Table 5.1: **Comparisons to Baselines.** Semantic segmentation (Class Accuracy and IoU), FID [51] and Perceptual Distance (PD) evaluations on different datasets for GVSNet (ours) along with SPADE+X baseline techniques, that first perform semantics-to-RGB conversion followed by monocular NVS.

to render novel views. For the latter task, we pick Stereo-magnification (SM) [149] adapted to single-view NVS, Continuous View Synthesis (CVS) [24], and Appearance Flow (AF) [150]. We refer to these baselines as 'SPADE+X' where 'X' could be 'SM', 'CVS' or 'AF'. Table 5.1 shows the quantitative results. Results show that GVSNet is consistently superior to the SPADE+X techniques across different metrics and datasets. This demonstrates that GVSNet can better preserve the semantic structures of the input while generating more realistic images. Figure 5.4 shows sample visual results. The visual results in Figure 5.1 (play the animation), Figure 5.4, and Figure 5.5 further validate that GVSNet better preserves semantic structures and geometric consistency. We show more visual results in the supplementary video at the project webpage [1].

Method	Class Acc. ↑	IoU ↑	PD ↓	FID ↓
GVSNet Variations				
SUN + SPADE [88]	72.92	65.52	1.75	68.96
SUN + LTN	71.90	63.12	1.83	69.46
SUN + LTN + ADN (Full model)	**74.34**	**66.43**	**1.74**	**62.06**

Table 5.2: **Ablative Studies on GVSNet.** Semantic segmentation (Class Accuracy and IoU), FID [51] and Perceptual Distance (PD) evaluations on CARLA [30] with different variations of our GVSNet. Results show that all the component networks of SUN, LTN and ADN are important.

Ablation Study. GVSNet comprises several computational blocks, each critical to its success. To confirm this, we perform an ablation study and evaluate the impact of swapping out parts of it. One basic test is to only use the semantic uplifting network (SUN) to synthesize 2D semantics in the target view, and use SPADE [88] to convert it to RGB image. We refer to this model as 'SUN+SPADE'. To evaluate the importance of f-dimensional layered appearance features (f=20) as opposed to just estimating 3-dimensional color,

[1] https://gvsnet.github.io

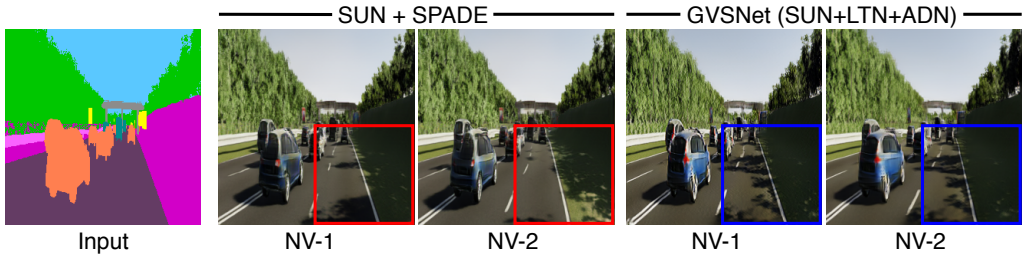


Figure 5.5: **Consistency across views.** We show sample generated images in two novel views (NV-1 and NV-2) using our full GVSNet model along with those obtained with SUN+SPADE. SUN+SPADE model first estimates novel view semantics followed by semantics-to-RGB translation with SPADE [88]. Results show that view-independent translation results in inconsistency across the views and the layered translation (LTN) is important to generate view-consistent images.

Method	CARLA [30]	Virtual-KITTI-2 [13]
GVSNet	0.0154	**0.0328**
SPADE [88]+SM [149]	**0.0130**	0.0330
SUN+SPADE [88]	0.0253	0.0511

Table 5.3: **Multiview Consistency** measured by warping two novel views onto a reference camera and calculating their difference as *mean-absolute-brightness* error. GVSNet is more consistent than SUN+SPADE. GVSNet is almost as consistent as SPADE+SM, despite the latter computing novel views by resampling the input view directly. qualitative results presented in Figure 5.5.

we experiment with a model that translates layered semantics to layered color images. We refer to this variant as 'SUN+LTN', as it does not use ADN. Ablation results shown in Table 5.2 indicate that all the three component networks (SUN, LTN and ADN) are important. Performance drops considerably if we use LTN to directly translate to color MPI (SUN+LTN vs. SUN+LTN+ADN). Table 5.1 also indicate that 'SUN+SPADE' performs better on segmentation metrics, while performing similarly on perceptual distance metrics.

We observe that SUN+SPADE, that applies SPADE [88] on projected 2D semantics in novel views, can generate photorealistic novel view images. However, we also notice that SUN+SPADE produces inconsistent images across multiple target views as image translation is applied independently on each target view semantics. Figure 5.5 shows a sample visual result indicating much better view-consistency with GVSNet when compared to SUN+SPADE model. Numerical evaluations in Table 5.3 also show that GVSNet is more consistent than SUN+SPADE. We argue that geometric or appearance inconsistency among different views is more detrimental than a slight perturbation of the semantic information, when this is common to all the synthesized views.

Semantic Editing. One of the main advantages of GVS is to simplify multi-view content

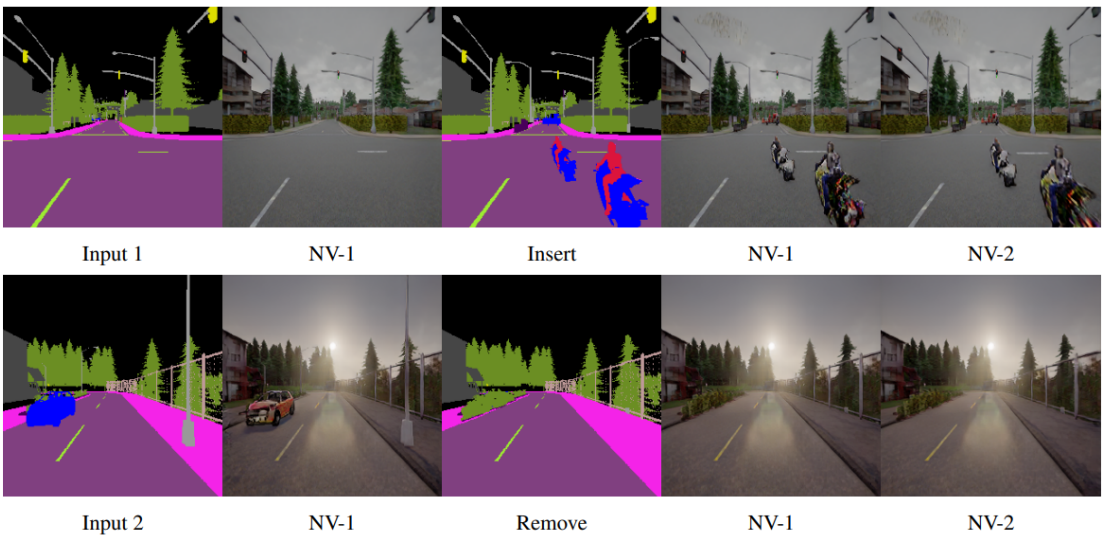


Figure 5.6: **Semantic Editing.** Sample GVSNet generated novel view (NV-1 and NV-2) images before and after editing objects in a given semantics. Results with both object insertion (left) and object removal (right) show that GVSNet can generate realistic images also on manually edited semantics.

generation. Editing a semantic map by simply pasting an object from another, for instance, is arguably easier than directly editing RGB pixels, which requires accounting for lighting conditions explicitly, adding realistic textures, etc. Similarly, despite the success of recent inpainting methods [77], removing objects from RGB images while accounting for the scene context requires advanced skills. Thanks to our method, simple object addition to/removal from the 2D semantic maps seamlessly translates to photorealistic multi-view images, as shown in Figure 5.6.

Depth from Semantics. Since we estimate opacity α at each MPI layer from a single 2D semantics, we can convert the MPI opacity into depth by performing alpha composition on the depth values of the MPI planes (see Equation 5.2 for details). Figure 5.7 and Figure 5.8 show sample depth estimations with our GVSNet that *only* takes 2D semantics as input. For comparison, we also show depth estimation from the RGB image using MonoDepth [40] (MD), which is trained in a fully-supervised manner using GT depths. Results indicate that our depth estimates are more accurate at thin structures. This is further confirmation of the strong structural cues offered by the semantics.

Our Semantic Uplifting Network (SUN) estimates multiplane opacities α, in the reference camera. Here we show a simple way to convert $\alpha \in R^{n \times m}$ into a depth map $\hat{Z} \in R^{n}$, where n is the number of pixels and m is the number of planes in the MPI representation. Suppose the MPI planes are located at distances $\{d_1, d_2, \ldots, d_m\}$ and are fronto-parallel to the reference camera. The depth value at a pixel p, $\hat{Z}(p)$, is computed by alpha compositing the depth of the MPI planes with the alpha values at pixel p, as follows.

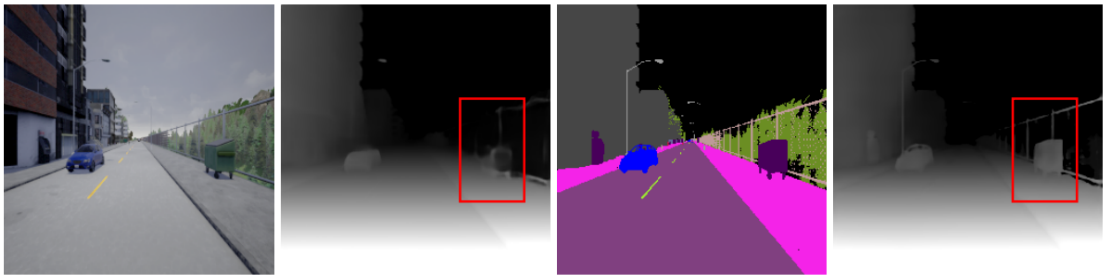


Figure 5.7: **Depth Estimation** We can obtain relative depth using the estimated MPI opacity from 2D semantics. Also shown is the result obtained with MonoDepth [40] (MD) that uses RGB image as input. Results show that GVSNet (ours) can estimate better geometric structures.

$$\hat{Z}(p) = \sum_{i=1}^{m} \left[d_i \, \alpha(p,i) \left[\prod_{j=1}^{i-1} (1 - \alpha(p,j)) \right] \right] \tag{5.2}$$

In Table 5.4, we show accuracy of the depths estimated by our SUN method and compare it against a baseline network trained with ground truth depths. The baseline network uses an RGB input while our method uses semantics as input. The baseline network is taken from the MonoDepth [40]; it is an encoder decoder network with ResNet-18 backbone. The network can be trained in a self-supervised manner, however for a fairer comparison, we train the network using ground-truth depth.

Method	Input	Supervision	SC_Inv	L1_Rel	L1_Inv	Depth range (in meters)
Ours	Sem	Sem + Depth	**0.200**	0.129	**0.006**	1 - 100
Mono-Depth	RGB	Depth	0.225	**0.125**	0.008	1 - 100
Ours	Sem	Sem + Depth	**0.252**	0.184	**0.006**	1 - 200
Mono-Depth	RGB	Depth	0.269	**0.155**	0.008	1 - 200
Ours	Sem	Sem + Depth	0.**331**	0.395	**0.006**	1 - 1000
Mono-Depth	RGB	Depth	0.358	**0.268**	0.007	1 - 1000

Table 5.4: **Depth accuracy evaluation on CARLA [30] dataset.** Depth maps generated by our SUN model and monocular depth estimation network from [40]. SC_inv, L1_Rel and L1_Inv stand for scale invariant, L1 relative and L1 inverse depth metrics, respectively.

We evaluated these networks following traditional depth accuracy metrics [130]: *scale invariant depth* (SC_Inv), *relative depth* (L1_Rel) and *inverse depth* (L1_Inv) metrics. Using only semantics as input, which is devoid of photo-metric details, our network produces favourable depth estimations when compared to the baseline. Our method outperforms

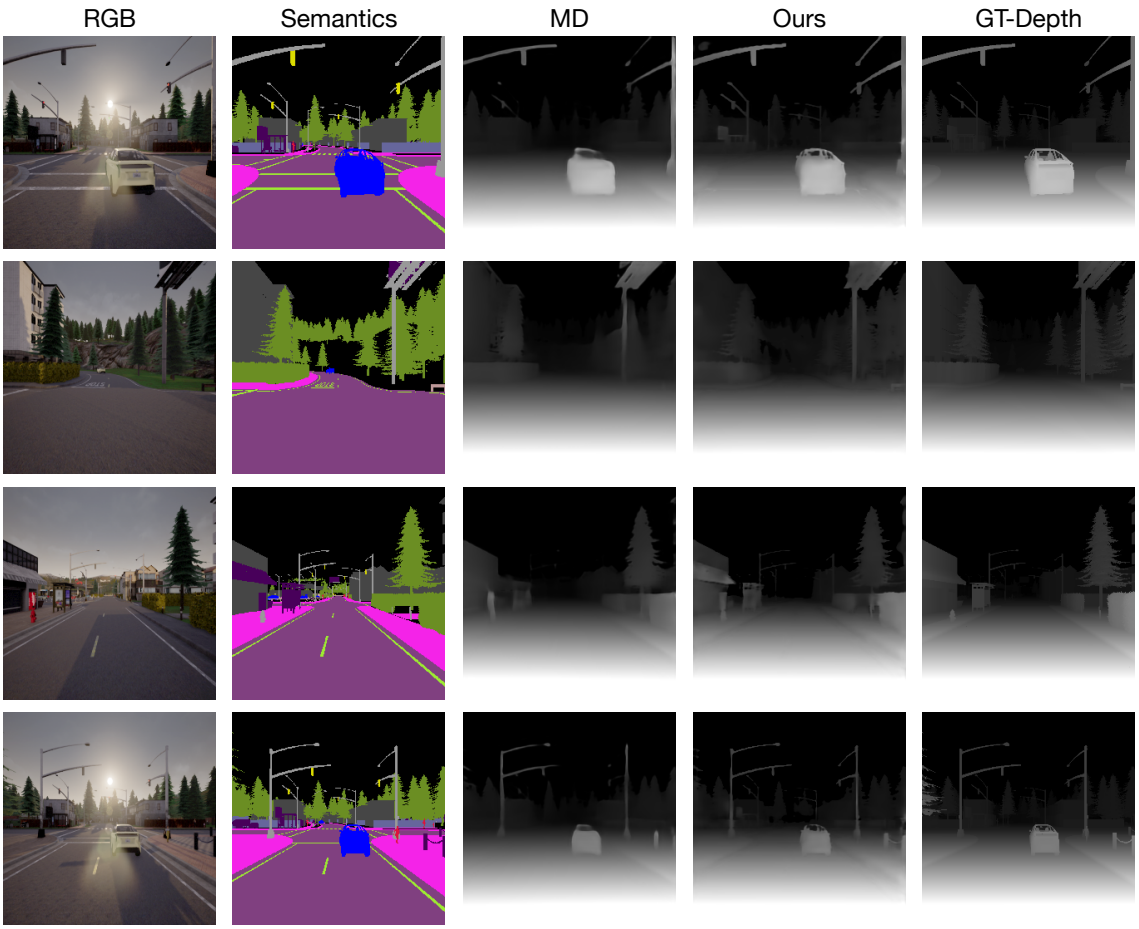


Figure 5.8: **Depth Estimation Results.** Here we present more visual depth estimation results from our SUN model and MonoDepth (MD) [40] model trained with ground-truth depths.

the MonoDepth network on SC_Inv and L1_Inv metrics. However, on L1_Rel, our method under-performs compared to MonoDepth. We believe this is because MPI planes are distributed by sampling the inverse depth linearly. This results in having few planes far from the camera. Nonetheless, for the view synthesis tasks it is desirable to have higher accuracy at closer ranges.

Comparison RGB+Seg View Synthesis Methods In our previous experiments we compared against baseline constructed as a sequential application of Image-to-Image Translation and Monocular Novel view Synthesis (MNVS). The results indicate that our method preserves the semantics/geometry of the input scene. In order to further improve the baselines, we experimented with feeding the MNVS part with RGB and Semantics. The RGB and Semantics are concatenated channel wise and fed as input to MNVS methods. As expected, feeding RGB and Semantics improves the performance of all baselines. Table 5.5

shows that the SPADE [88]+SM [149], baseline that takes an additional semantic map (indicated as SPADE [88]+SM [149]$^{+}$) performs better than its RGB counterparts.

Method	Cls. Acc ↑	IoU ↑	PD↓	FID↓
GVSNet	**74.34**	**66.43**	**1.74**	**62.06**
SPADE+SM	69.93	60.82	1.95	75.81
SUN+SPADE	72.92	65.52	1.75	68.96
SPADE+SM^{+}	72.29	63.55	1.75	73.74

Table 5.5: **Comparing against an SPADE [88]+SM [149]** baseline, where the SM network takes a translated image together with the input semantics. As show in rows 3 and 4 the feeding the SM network with semantics leads to better performance. However, as can be seen in the first column our method consistently outperforms the improved and the original baselines.

Evaluations with Different Style Images. As can be seen in the Figures 5.1 and 5.4, our method can produce novel views with different styles, as dictated by the given style images. In image-to-image literature [88], it is customary to use the target color image as a style during evaluation. Following this custom, we use the color image from the source camera as style guidance for the novel view image generation. In order to verify the efficacy of method under arbitrary styles, we perform an additional test where our method is not fed the input image as style, rather a random frame from the same sequence is used as style input. In Table 5.6, we show semantic evaluation results using this procedure. The results show that our method still outperforms the baselines under this setting of using different style image.

Method	Cls. Acc. ↑	Mean IoU ↑
GVSNet Variations		
GVSNet (Full Model)	**73.16**	**65.40**
SUN+SPADE [88]	70.95	63.73
SPADE [88] + SM [149]	67.26	58.71
SPADE [88] + CVS [24]	64.70	55.67
SPADE [88] + AF [150]	63.28	54.05
Target GT Images	77.47	69.67

Table 5.6: **Semantic evaluations on CARLA [30] dataset.** Metrics are computed using a single randomly chosen style frame per sequence. This test shows that our method handles arbitrary style images. Our Class Accuracy and Mean IoU results here are also close the results achieved using the source-view color image as style input.

5.5 Conclusion

In this chapter, we propose Generative View Synthesis Network (GVSNet), which produces photorealistic novel view images from only a single 2D semantic map. The proposed method grants users control over the 3D scene, by allowing users to edit the scene geometry and appearance. Moreover, the method provides users a means to create novel scenes by painting a single view as a 2D semantic map. We demonstrate that the simple application of existing techniques for this problem yields inadequate results. Our key insight is to leverage the structural information in the input semantics by uplifting the 2D semantics to layered 3D semantics. Further, we carry the structural 3D semantic information forward with a layered semantics-to-RGB translation network. Comprehensive experimental analysis on three different datasets demonstrate the potential of GVSNet in generating geometrically consistent novel view images, while preserving the structures in the input semantics.

5.6 Additional Experimental Details

5.6.1 Dataset Details

We use three publicly available datasets for our GVS experiments. In all 3 datasets we used images down scaled to a resolution of 256×256 pixels.

CARLA. Using the CARLA [30] open source simulation environment, we captured 20 independent sequences in 5 towns. We use 16 sequences for training and 4 sequences for testing, taking one test sequence per city. We capture data using camera arrays mounted on top of a car. Each camera captures color, semantic and depth images. We use 3 groups of camera arrays: *horizontal*, *forward* and *side*-camera groups. The horizontal camera array has 5 cameras at uniform spacing along the $x - axis$. The forward camera array has 5 cameras uniformly distributed along $z - axis$. The side camera array contains 3 horizontally shifted cameras facing the side-view of the car. The spacing between cameras within each array is $54cms$. During training and testing we take a random pair of cameras from one of the camera groups and use one as source and the other as target.

Cityscapes is a publicly available dataset of urban scenes captured across several German cities [28]. We use 2975 scenes for training and 500 scenes for test. Each scene is captured with a stereo pair. During training and testing phases, we use one of the two stereo cameras as source and the other as target. Ground truth semantics is available for the left camera images only. We label the right camera images using a pre-trained semantic segmentation network [151]. Cityscapes dataset has no ground-truth depth. We generate depth maps by training the DPS [60] network in a self-supervised manner. In order to achieve results which are comparable with SPADE [88], we use instance masks in our experiments on this dataset. We compute a one channel instance mask image, as a gradient of the original instance segmentation. We use instance masks by concatenating the input view instance mask with lifted semantics. Since right camera instance masks are not available, we generate a pseudo ground-truth instance masks by warping the left

image masks using the depths estimated by DPS network [60]. While warping, we perform forward-backward depth tests to detect occlusions and in areas where occlusion is detected we leave the instance masks empty. Note, that in other datasets we do not use instance masks. In all of our experiments, we use 19 class labels provided by the dataset.

Virtual-KITTI-2 is a synthetic dataset of urban scenes captured with a stereo camera. Each stereo pair has color images together with ground-truth depths and semantic segmentations. The dataset has 6 sequences captured under 5 weather conditions. Each sequence is randomly divided into train and test sub-sequences. The training sub-sequence covers 80 % of the frames and the test sub-sequence covers 20% of the frames.

5.6.2 Training Losses

Depth Loss. We compute the depth reconstruction loss $\mathcal{L}_{dep}$ on a scaled version of the predicted and target inverse depth maps. The scaled inverse depth is computed as $f_x \times 0.54/depth$, where f_x is the focal length. This is equivalent to converting *depth* into *disparity* by assuming imaginary stereo camera with a baseline of $54cms$. Since, all of the datasets used in this chapter are large scale outdoor scenes, the same scaling works well for all the datasets.

Loss Weighting. We set the depth loss weighting factor $\lambda_1 = 0.1$, while the other weighting factors (λ_0, λ_2 and λ_3) are all set to 1.

Training Protocol GVSNet is trained in two phases. In the first phase, the Semantic Uplifting Network (SUN) is trained using depth reconstruction and semantic alignment losses. For the CARLA [30] dataset, we train the SUN network for 30 epochs. In Cityscapes [28] dataset, we train for 200 epochs and in Virtual KITTI-2 [13], we train for 60 epochs. In all datasets, we use mini-batch size 12 and Adam [69] optimizer with a $lr = 0.0004$, $\beta_1 = 0.900$, $\beta_2 = 0.999$), and $eps = e{-}08$. This SUN training is performed using 3 NVIDIA GTX-2080-Ti GPUs.

In the second phase, the Layered Translation Network (LTN) and Appearance Decoder Network (ADN), are trained by minimising the color and GAN losses. We train these networks while keeping the SUN network fixed. For CARLA dataset, we train for 20 epochs. In Cityscapes and Virtual KITTI-2 datasets, we train for 250 and 35 epochs respectively. In this phase, we use a batch size of 16. The training is done using 8 NVIDIA GTX-2080-Ti GPUs. We use Adam [69] optimizer with the initial learning rate of $lr = 0.0004$. The learning rate is kept fixed for the first half of the training. In the second half, we start decreasing the learning rate linearly so that it reaches 0 at the last iteration.

5.6.3 Network Architecture

Semantic Uplifting Network. Our Semantic Uplifting Network (SUN), is a 2D encoder-decoder network with 3 outputs: lifted semantics, MPI opacity α and association function Φ. These 3 outputs are predicted jointly, with the network architecture shown in Table 5.7 In

Layer	Input	Type	Stride	in_chans	out_chans
conv1a	input sem	Conv2d + ReLU	2	1	32
conv1b	conv1a	Conv2d + ReLU	1	32	32
conv2a	conv1b	Conv2d + ReLU	2	32	64
conv2b	conv2a	Conv2d + ReLU	1	64	64
conv3a	con2b	Conv2d + ReLU	2	64	128
conv3b	conv3a	Conv2d + ReLU	1	128	128
conv4a	conv3b	Conv2d + ReLU	2	128	256
conv4b	conv4a	Conv2d + ReLU	1	256	256
conv5a	conv4b	Conv2d + ReLU	2	256	512
conv5b	conv5a	Conv2d + ReLU	1	512	512
conv6a	conv5b	Conv2d + ReLU	2	512	512
conv6b	conv6a	Conv2d + ReLU	1	512	512
conv7a	conv6b	Conv2d + ReLU	2	512	512
conv7b	conv7a	Conv2d + ReLU	1	512	512
dconv7	conv7a	Conv2d + ReLU	1	512	512
dconv6	dconv7 $\bigoplus$ conv6b	Conv2d + ReLU	1	1024	512
dconv5	dconv6 $\bigoplus$ conv5b	Conv2d + ReLU	1	1024	512
dconv4	dconv5 $\bigoplus$ conv4b	Conv2d + ReLU	1	768	384
dconv3	dconv4 $\bigoplus$ conv3b	Conv2d + ReLU	1	512	256
dconv2	dconv3 $\bigoplus$ conv24	Conv2d + ReLU	1	320	96
dconv1	dconv2 $\bigoplus$ conv1b	Conv2d + ReLU	1	128	96
base_1	dconv1	Conv2d + ReLU	1	96	96
base_2	base_1	ResBlock	1	96	96
base_3	base_2	ResBlock	1	96	96
base_4	base_3	ResBlock	1	96	96
base_5	base_4	ResBlock with BN	1	96	96
base_6	base_5	ResBlock with BN	1	96	96
out_conv_1a	base_6	Conv2d+ReLU	1	96	(l x (k-1) + m*(k+1))/2
out_conv_1b	out_conv_1a	Conv2d+ReLU + BN	1	(l x (k-1) + m*(k+1))/2	(l x (k-1) + m*(k+1))/2
out_conv_a	out_conv_1b	Conv2d+ReLU	1	(l x (k-1) + m*(k+1))/2	(l x (k-1) + m*(k+1))/2
out_conv_1b	out_conv_a	Conv2d	1	(l x (k-1) + m*(k+1))/2	l x (k-1) + m*(k+1)

Table 5.7: **Semantic Uplifting Network Architecture.**

Table 5.7 *in_chans* and *out_chans* refer to input and output number of channels, respectively. The size of the SUN network depends on the following hyper-parameters: number of layered semantic maps (k), number of MPI planes (m) and number semantic classes (l).

Layer	input	in_chans	out_chans
conv_0	appearance features	f	16
conv_1	conv_0	16	32
conv_2	conv_1	32	64
conv_3	conv_2	64	64
conv_4	conv_3	64	64
d_conv_4	conv_4	64	64
d_conv_3	d_conv_4 $\bigoplus$ conv_3	128	32
d_conv_2	d_conv_3 $\bigoplus$ conv_2	64	32
d_conv_1	d_conv_2 $\bigoplus$ conv_1	64	16
output_conv	d_conv_1 $\bigoplus$ conv_0	32	3

Table 5.8: **Appearance Decoder Network Architecture.**

The output from the *out_conv_1b* layer is split into α, Φ and layered semantics channels. α and Φ take up m and $k \times m$ channels, respectively. Since we create the layered semantic representation includes the input semantics, the network predicts layered semantics only for $(k-1)$ layers, with a total of $(k-1) \times l$ channels. Sigmoid activation is applied on the α and Φ outputs. All of the layers in this network are 2D convolutional layers. Encoder layers $conv1a$ up to $conv7b$ decrease the spatial resolution of their output by a factor of 2 using stride 2 convolutions. Decoder layers $dconv7$ to $deconv1$ apply nearest neighbour up-sampling with a factor of 2 before applying convolution and ReLU. The ResBlock have 2 convolutional layers (the first one has ReLU activation), and the output of the second layer is added to the input. The $\bigoplus$ sign refers to channel-wise concatenation.

Appearance Decoder Network. Table 5.8 shows the detailed architecture of the Appearance Decoder Network (ADN) network. In Table 5.8, in_chans and out_chans stand for the number of input and output channels. The $\bigoplus$ sign indicates channel-wise concatenation. The ADN network gets $f-channel$ appearance feature as input and returns a $3-channel$ color image as output. Every layer in this network is a convolutional layer with 3×3 kernel and *stride* of 1. All layers except output_conv have spectral normalisation and LeakyReLU non-linearity (with negative slope value of -0.2) after convolution. output_conv layer applies Tanh non-linearity after convolution and it has no spectral normalisation [84]. Layers conv_0 up to conv_4 down scale the spatial dimensions of their output using a bilinear down-sampling kernel. Equivalently, layers d_conv_4 up to d_conv_1 perform bilinear up-sampling with a factor of 2, before convolution is applied.

Layered Translation Network. Layered Translation Network (LTN) consists of layered appearance generator and style encoder networks. The architecture of both the appearance generator and encoder networks are similar to those used in the SPADE [88] paper. However, in this chapter the generator network performs layered translation of k semantic maps. The output of the network is also different since we predict higher dimensional features, not color images. Our generator, takes input semantics of shape $[H \times W \times k \times l]$ and

produces appearance features with dimensions $[H \times W \times l \times f]$. In our experiments, we found that $k = 3$ and $f = 20$ suffice to achieve good results. The GAN part of our work is based on SPADE, therefore, we use the same discriminator network and training losses as in the SPADE work.

6 Learning to Generate Multiplane Image Scene Representations from Text

In the previous chapters, we investigated view synthesis approaches with a wide range of input complexities and types. In chapter 5, we presented the GVSNet approach, which requires a single 2D semantic map as input. Despite the great creative freedom it offers, GVSNet has limited usability as creating and/or editing semantic maps is a tedious task. Therefore, in this chapter we alleviate this limitation by further simplifying the required input from a 2D semantic map to a textual description. Effectively reducing the time required to generate or edit a scene from minutes to seconds.

The work presented in this chapter is taken from the following article:

1. Tewodros Amberbir **Habtegebrial**, Stefan Popov, Varun Jampani, Didier Stricker and Vittorio Ferrari
   Text2MPI: Learning to Generate Multiplane Images from Text. Under Review.

Contents

6.1 Introduction **102**
6.2 Related Work **104**
6.3 Proposed Method **105**
6.4 Experiments **111**
6.5 Conclusion **118**

6.1 Introduction

In the previous chapters, we have addressed the Image-based Rendering (IBR) problem from progressively simpler input settings, such as:

- Multi-view Stereo,
- Unstructured Spherical Light Fields, and
- 2D Semantic Maps.

The GVSNet approach, proposed in Chapter 5, introduces a method of performing IBR from 2D semantic maps. In addition to requiring minimal user input, it also offers control over geometry and appearance of the scene being rendered. However, GVSNet still expects the user to create 2D semantic maps, or edit existing ones, which could easily become a tedious task. In this chapter, we aim to push the envelope even further, by simplifying the required input to: a *textual description* of the desired scene. To this end, we propose Text2MPI, a latent diffusion model [101] for generating renderable Multiplane Images (MPI) [149] representations of complex scenes, from given textual descriptions.

Text-to-Image models, especially diffusion [117, 101] based ones, are pushing the limits of visual content creation. With the rise of commercial, ready-to-use solutions such as Dall-E [96], these models have gained widespread adoption. General purpose text-to-image models such as LatentDiffusion [101], Imagen [105], and Dall-E [96], have been extended further to tackle challenging computer vision and graphics problems such as video generation [52].

Recent works have also pushed diffusion based generative models into the realm of 3D content creation. This enables digital artists to convert their ideas into 3D assets using simple textual descriptions. Existing works in this field employ a variety of strategies, such as score-distillation-based NeRF optimization with a pre-trained 2D-model [92], tri-plane optimization [14], and diffusion in 3D on a feature grid [66]. Yet, despite their immense potential, existing works are mostly limited to single object scenes.

In this chapter, we aim to generate renderable 3D representations of complex real world scenes, from textual descriptions only. We propose *Text2MPI*, a learned model that maps input text descriptions to Multiplane Images (MPI) [149], and we show how to train a hash-based [85] 3D representation on top, to allow larger camera movements. Our 3D representation allows rendering of high quality realistic views of complex scenes, with no inconsistencies among them, and it allows users to not only generate images, but to also change the viewpoint of the generated scenes at will.

An MPI is a layered representation that uses a set of RGBα textures, mapped to pre-determined parallel planes in 3D space. MPIs are encoded as dense tensors with regular shape, which in principle allows learning generative diffusion models [101, 105] over them. In practice, MPIs require high resolutions and large number of planes for good results, which makes them memory-intensive. Standard MPIs are prohibitively expensive when combined with a diffusion model, both in terms of memory and computation. To address

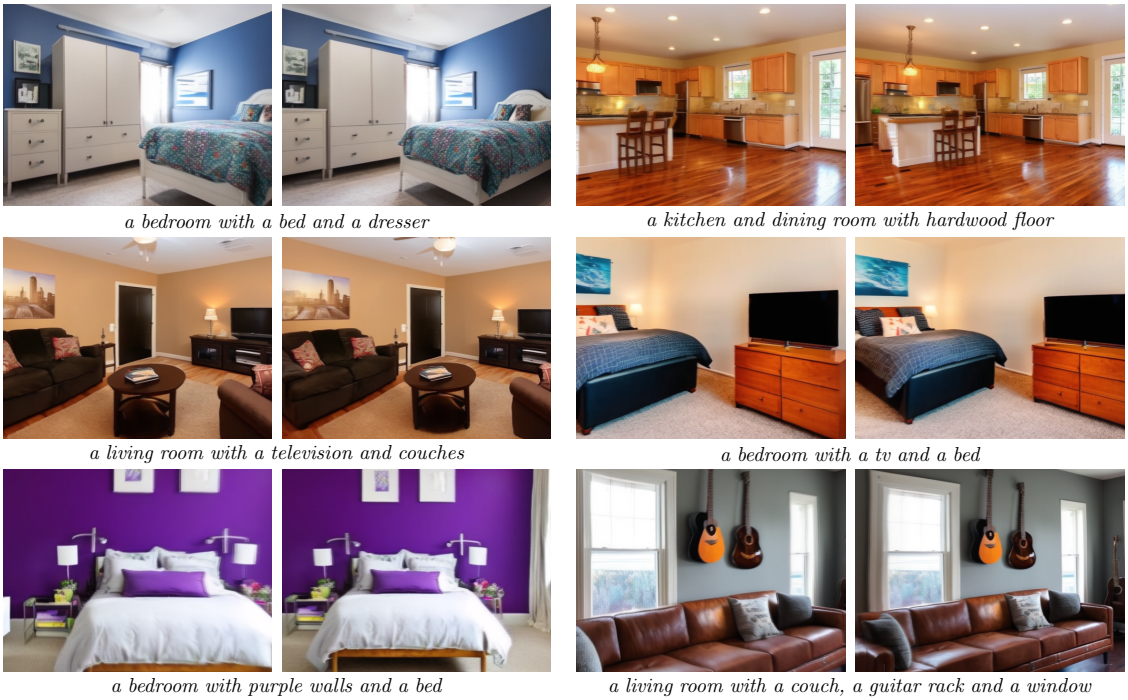


Figure 6.1: **Text2MPI** generates renderable 3D representations from an input textual description of a desired scene. We show pairs of views rendered from such representations, with the input text below them. Notice the high level of realism and the near absence of artifacts in all views.

this, we introduce a compact latent MPI representation (Section 6.3.1), which reduces MPI's memory requirements by a factor of 256, without noticeable loss in quality. This in turn reduces the memory complexity of the diffusion model and increases its performance by the same factor.

Text-to-image diffusion models [53, 117, 101, 105, 52] are commonly trained on large corpora of images, paired with corresponding textual descriptions. In contrast, our method trains on multi-view data (on RealEstate10K [149]), which is much smaller, covers only a narrow domain of the visual world, and has no textual descriptions. To leverage the generalization capability and knowledge of pre-trained image diffusion models, we propose a mechanism for guiding our Text2MPI method with image diffusion models (Section 6.3.3). Similar to [94], we jointly generate both the image at the reference view of the MPI, as well as the MPI itself. At each diffusion step, the image diffusion model [101] generates a conditioning signal for our MPI diffusion model. This allows us to train Text2MPI models that generalize to objects and concepts not present in the multi-view training dataset (see examples in Figure 6.9).

We perform experiments on the RealEstate10K dataset [149] (Section 6.4). We observe that our method can generate 3D representations with complexity matching the scenes in the dataset. In addition, views rendered from our 3D representations are high-quality,

sharp, visually appealing, realistic, and consistent among each other. We also observe strong generalization abilities, where Text2MPI is able to generate scenes with objects and concepts that are not present in the multi-view training set. Compared to state-of-the-art single-image Novel View Synthesis methods, views rendered from Text2MPI-generated representations have higher realism scores.

Contributions. Our contributions can be summarized as follows:

- We propose a mechanism for training diffusion models over MPI representations.
- To make training feasible, we propose an approach that compresses MPIs to a significantly smaller latent representation.
- We propose a mechanism to transfer the vast generalization abilities of text-to-image models into our MPI diffusion model.
- Finally, we show how to train InstantNGP [85] on top of our model's outputs, to enable larger camera movements during view synthesis.

The rest of the paper is organized as follows: In section 6.2, we present prior works in generative models and Image-based Rendering. Section 6.3, introduces our latent MPI representation, the MPI diffusion model, our image-model-based guiding mechanism, and our method for training hash-based 3D representations from MPIs. Section 6.4 covers the setup and results of our experiments. Section 6.5 concludes our findings.

6.2 Related Work

In this section we present a brief overview of prior works on image-based rendering (IBR) and generative models for visual content synthesis.

Image-based Rendering: The history of Image-based Rendering (IBR) works predates the modern neural rendering era. The earliest works in IBR include image-space warping [22], light fields [73], and Lumigraphs [43, 12]. The advancements in multi-view stereo reconstruction algorithms have led to accurate and scalable IBR methods such as [16]. These methods leverage a proxy scene geometry acquired with multi-view 3D reconstruction or Structure-from-Motion (SfM).

A neural approach for Image-based Rendering (IBR) was first introduced in *DeepStereo* by Flynn [36]. DeepStereo used plane-sweeping and convolutional feature matching networks for accurate IBR and 3D inference. Zhou [149] further pushed the boundary of IBR by introducing Multiplane Images, an efficient scene representation and learning framework for accurate Novel View Synthesis. The introduction of Neural Radiance Fields (NeRF) [83] has ignited unprecedented interest and progress in Image-based Rendering. NeRF uses coordinate-based Multi-Layer Perceptrons to represent 3D scenes.

Followup works have extended NeRFs for scenarios such as dynamic scenes [93], large-scale urban scenes [122, 100], scene with reflective objects [131], scene relighting [118], long range tracking [132], multi-view stereo [18], and simultaneous localisation and mapping [103].

Neural Approaches for Visual Content Creation: Semantic image synthesis (SIS) [88] and image-to-image translation (I2I) [62, 133] works introduced novel generative models that enable users to create visual content. These models leverage Generative Adversarial Networks [42] that were trained to solve visual analogy [50] problems. SIS methods were extended towards generating 3D-consistent multiviews from single 2D semantic maps [3, 59, 146].

In recent years, Diffusion Models [117, 53] have led breakthroughs with text-to-images model such as LatentDiffusion models [52, 105, 96]. Moreover, new techniques such as DreamFusion [92], Dreambooth3D [95], HoloDiffusion [66] have introduced text based 3D generative models. Diffusion generative models have also been used for text-and-depth guided mesh texturing [124], HDR panoramas for photorealistic lighting [25], room reconstruction [54] etc. Majority of works in this area are geared towards single object reconstruction. In this chapter we strive to learn a generative model that produces an MPI [149, 128] representation of complex real-world scenes from text.

6.3 Proposed Method

Given a textual description, our method *Text2MPI* generates a 3D representation of a matching scene, from which we render high quality, 3D-consistent views of the scene. At the core of our architecture is a latent multiplane image representation (Section 6.3.1). We propose an encoder, to build this representation from input views, and a decoder, to obtain a full-resolution MPI [149] suitable for rendering views. To generate latent MPIs, we propose a modified latent diffusion model [101] (Section 6.3.2). We guide our model with a pre-trained image diffusion model [101] in order to improve its generalization abilities (Section 6.3.3). Finally, we extend the possible camera motion, by building InstantNGP [85] models from the decoded full-resolution MPI (Section 6.3.4).

6.3.1 Latent Multiplane Images

Preliminary. We start with a brief introduction on the multiplane image (MPI) [149] model[1]. An MPI is a renderable 3D scene representation, stored as textures on a set of d planes, parallel to a reference camera r and at predefined depths $\{z_i\}_{i=1}^{d} \in [z_{\text{near}}, z_{\text{far}}]$. Pixels in the $h{\times}w{\times}4$ textures contain R, G, B, and transparency α.

MPIs are built from a fixed number of views, n, of a static scene, each with resolution $h{\times}w{\times}3$. One of the input views is chosen as the reference r and the rest are re-projected

[1]For a comprehensive review of the Multiplane Image model, please refer to Chapter 2

onto it at the d fixed depth planes, forming in this way a $n{\times}d{\times}h{\times}w{\times}3$ plane sweep volume [7]. A neural network converts this volume into a $d{\times}h{\times}w{\times}4$ MPI. Novel views are computed again through homography, warping the textures on the d planes to the novel view, followed by back-to-front alpha composition.

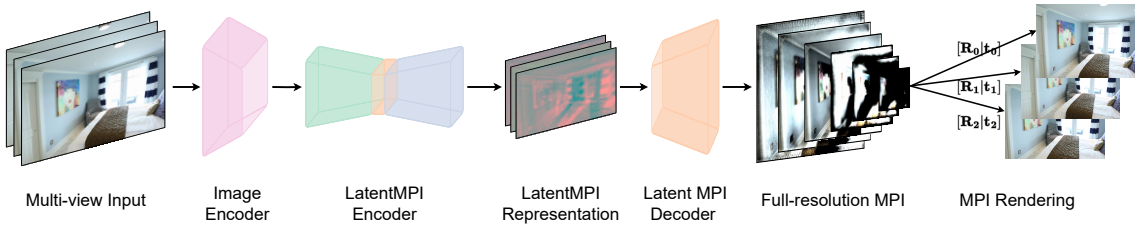


Figure 6.2: **An overview of the latent MPI model.** We encode the input images in latent space, we build a plane sweep volume through homography, and we build the latent MPI representation with a neural network. To render novel views with given camera rotations and translations $[R|t]$, we first decode the latent MPI into a full-resolution one.

Latent MPI. Training a diffusion model over MPI representations can be prohibitively expensive, due to the large number of MPI planes, coupled with the high resolutions required for visually appealing results. To address this, we propose a latent MPI model in the spirit of latent images [101]. An overview of the latent MPI model is provided in Figure 6.2.

We store latent MPIs in a $\frac{h}{k}{\times}\frac{w}{k}{\times}f$ feature array directly, rather than in d textures each with shape $h{\times}w{\times}4$. We build latent MPIs in a similar way to standard MPIs, using d planes parallel to the reference view, and we decode them into standard $d{\times}h{\times}w{\times}4$ MPIs when rendering novel views.

For our implementation, we chose $k = 8$, $d = 64$, and $f = 64$, reducing the number of elements by a factor of 256, compared to a full-resolution MPI representation. In turn, this reduces the memory and computation requirements of the downstream MPI diffusion task by the same factor.

Latent MPI encoder. To build a latent MPI, we encode each input image into a $\frac{h}{k}{\times}\frac{w}{k}{\times}4$ latent representation, using the pre-trained image encoder from [101], which reduces image resolution by the same factor as us ($k = 8$). Similar to standard MPIs, we build a $n{\times}\frac{h}{k}{\times}\frac{w}{k}{\times}4$ plane sweep volume at the reference view through homography. We use a UNet-style neural network to encode the volume into a latent MPI. Following [101], we adopt a variational approach, where we predict f-dimensional mean and variance for each pixel. We sample a latent MPI with the re-parameterization trick [70] during training and we take the mean during inference.

Latent MPI decoder. To render views, we first decode the latent $\frac{h}{k}{\times}\frac{w}{k}{\times}f$ MPI into a full-resolution $d{\times}h{\times}w{\times}4$ MPI, using a convolutional neural network. We then render views from this MPI in the standard way, through homography and alpha compositing [149].

Training the encoder and the decoder. We train the neural networks in the encoder and the decoder jointly, on multi-view data. For each scene, we sample $n = 3$ input views and one target view. We build a latent MPI from the input views using the encoder, we decode the latent MPI into a standard one, and we render an image for the target view. We minimize a set of photometric, geometric, and GAN losses, computed between the predicted and the ground-truth images for the target view:

$$\mathcal{L} = \mathcal{L}_{col} + \lambda_1 \mathcal{L}_{lpips} + \lambda_2 \mathcal{L}_{dep} + \lambda_3 \mathcal{L}_{gan} + \lambda_4 \mathcal{L}_{v_reg}$$

The color loss $\mathcal{L}_{col}$ measures L1 difference between images; $\mathcal{L}_{lpips}$ is a perceptual color loss [145]; $\mathcal{L}_{gan}$ is a combination of GAN hinge loss and discriminator feature reconstruction loss [88]; l_{dep} is a scale and shift invariant depth loss [98], between depth predicted by the MPI at the reference view and pseudo-ground-truth depth generated from DPT [97]; Following [101], we also apply a KL-Divergence regularization on the predicted latent MPI mean and log variance. In our experiments we set $\lambda_1 = 10$, $\lambda_2 = 20$, $\lambda_3 = 1$, and $\lambda_4 = 10^{-6}$. We train our model with the AdamW [78] optimizer with a fixed learning rate $lr = 5 \cdot 10^{-5}$, and we use $\beta_1 = 0.5$ and $\beta_2 = 0.9$.

6.3.2 Latent MPI Diffusion Model

Preliminary. [2]Diffusion Models (DMs) [53, 117, 101] are latent variable generative models, with an encoder-decoder structure. The encoder is fixed and gradually adds Gaussian noise to the data $\mathbf{x}_0$, according to a pre-defined variance schedule $\{\beta_t\}, t \in [1, 2, \dots T]$:

$$z_t = \sqrt{\bar{\alpha}_t}\, \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon_0 \tag{6.1}$$

where $\epsilon_0 \sim \mathcal{N}(0, I)$, $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$, $\alpha_t = 1 - \beta_t$.

The decoder predicts the noise ϵ_0 added to its inputs. It is usually implemented as a neural network $\mathcal{P}(\mathbf{z}_t|t)$ conditioned on time and often on text C: $\mathcal{P}(\mathbf{z}_t|C, t)$. Diffusion models are trained [53] by adding noise to the inputs (Equation 6.1) and then minimizing the difference to the noise predicted by the decoder:

$$||\epsilon_0 - \mathcal{P}(\sqrt{\bar{\alpha}_t}\, \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon_0|t)||^2 \tag{6.2}$$

During inference a diffusion model gradually maps a noise sample $\mathbf{z}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, into a clean sample, $\mathbf{z}_0$, by sampling from the distribution $p(\mathbf{z_{t-1}}|\mathbf{z_t})$ for time steps $T, T-1, \dots, 1$:

[2]We start this section by providing a brief overview of Diffusion Models and establishing necessary notations. For a more comprehensive review on Diffusion Models, please refer to Chapter 2.

$$\mathbf{z}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{z}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \mathcal{P}\left(\mathbf{z}_t | t\right) \right) + \sigma_t \, \epsilon$$
$$\text{where } \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{6.3}$$

Latent MPI diffusion model. We generate latent MPIs with a diffusion model, conditioned on text. To this end, we modify the first and the last UNet layers of a 2D latent diffusion model [101] to work with $\frac{h}{k} \times \frac{w}{k} \times f$ tensors, instead of $\frac{h}{k} \times \frac{w}{k} \times 3$.

For each visited scene during training, we sample $n = 3$ views and we build a latent MPI $\mathcal{Z}$, as described in Section 6.3.1. We add noise to $\mathcal{Z}$ (Equation 6.1) and we train our model $\mathcal{P}(\mathcal{Z}_t | C, t)$ to predict this noise. Our model is also conditioned on a textual description C of the scene.

We follow the standard procedure (Equation 6.3) at inference time. We gradually denoise a pure noise sample $\mathcal{Z}_T$ with our model, conditioned on a textual description C of the scene. We then convert the generated latent MPI $\mathcal{Z}_0$ into a full-resolution one (Section 6.3.1) to render views of the scene.

To train our model, we use the AdamW [78] optimizer with learning rate $lr = 10^{-5}$, and $\beta_1 = 0.5, \beta_2 = 0.9$. We apply an initial warm-up learning rate schedule for 10000 steps, as well as exponential moving average [125] to the model weights.

6.3.3 Generated 2D Prior

Diffusion models [53, 117, 101, 105, 52] are commonly trained on large corpora of images paired with corresponding textual descriptions. To leverage their knowledge, we designed our latent MPI diffusion model with few differences to LDM [101], and we initialize training from a pre-trained image diffusion checkpoint.

Nevertheless, we train on a much smaller multi-view dataset and thus risk catastrophic forgetting [144, 104, 56]. To overcome this, we guide our model with a pre-trained image diffusion model. We jointly generate both the latent MPI representation $\mathcal{Z}$ and the latent image $\mathcal{I}$ that matches the reference MPI view. Similar to [94], we use two inter-dependent towers to generate $\mathcal{I}$ and $\mathcal{Z}$ (Figure 6.3). The first one is a pre-trained image latent diffusion model $\mathcal{P}_{LDM}(\mathcal{I}_t | t, C)$ [101]. The second one is our latent MPI diffusion model $\mathcal{P}(\mathcal{Z}_t | C, t, \psi(\mathcal{I}_t))$, additionally conditioned to depend $\psi(\mathcal{I}_t)$ at every time step, where

$$\psi(\mathcal{I}_t) = \frac{1}{\sqrt{\bar{\alpha}_i}} \left(\mathcal{I}_t - \sqrt{1-\bar{\alpha}_t} \, \mathcal{P}_{LDM}(\mathcal{I}_t, t, C) \right). \tag{6.4}$$

During inference, we start from pure noise for both $\mathcal{Z}_T$ and $\mathcal{I}_T$. We then apply the image model $\mathcal{P}_{LDM}(\mathcal{I}_t, t, C)$ to generate $\mathcal{I}_{t-1}$ from $\mathcal{I}_t$. We compute $\psi(\mathcal{I}_t)$ with Equation 6.4, we condition our MPI model on it, $\mathcal{P}\left(\mathcal{Z}_t | C, t, \psi(\mathcal{I}_t)\right)$, and we generate $\mathcal{Z}_{t-1}$ from $\mathcal{Z}_t$; see Figure 6.3 for a graphical illustration. During training, we build an MPI representation from the input views for $\mathcal{Z}$ as usual (Section 6.3.2), and we take the reference view for

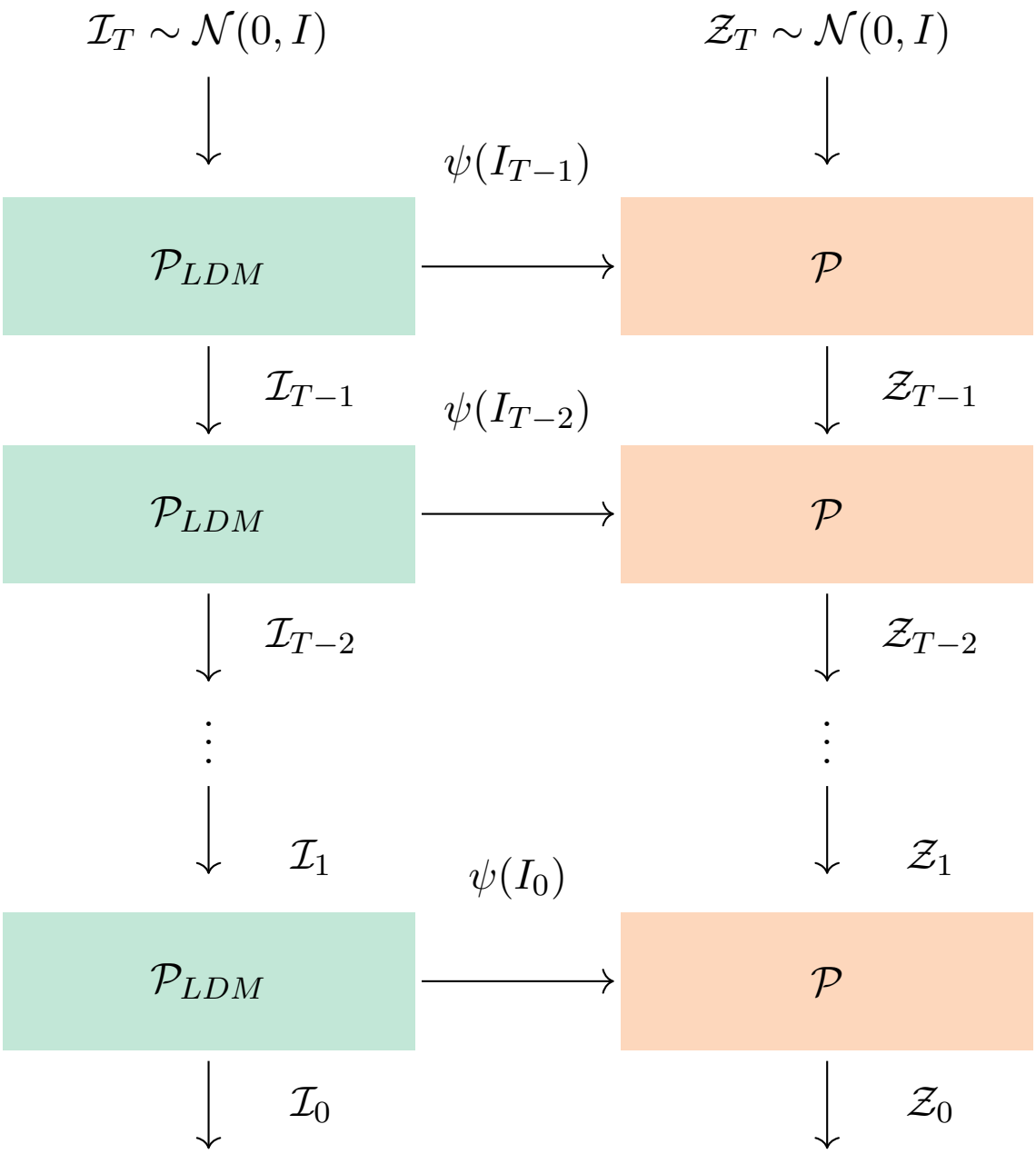


Figure 6.3: **Generated 2D Prior.** We jointly generate both the latent MPI representation $\mathcal{Z}$ and a latent image $\mathcal{I}$ that matches the reference MPI view. We condition the MPI model on the generated image with a 2D prior $\psi(\mathcal{I}_t)$. Both models are further conditioned on the time step and a text caption, which we omit here for brevity.

$\mathcal{I}$. We add noise to both (Equation 6.2) and we compute the 2D prior $\psi(\mathcal{I}_t)$ with the 2D model (Equation 6.4). In summary, we give our model the following inputs: noisy latent

MPI $\mathcal{Z}_t$, time step t, text caption C and 2D scene prior $\psi(\mathcal{I}_t)$. Our model predicts the noise added to $\mathcal{Z}_t$. We train only the latent MPI model, minimizing again the difference to the ground-truth noise (Equation 6.2).

6.3.4 Improving Scene Geometry and Extending Camera Motion

In spite of the many practical advantages they offer, MPIs suffer from strong limitations emanating from their discrete plane-centric nature. MPIs allow limited camera motion and often they produce depth maps that are blurry and highly spurious in areas with strong reflective properties. The discrete MPI scene representation leads to aliasing artifacts due to the Nyquist sampling criterion. Those aliasing effects become stronger as the magnitude of the camera translation increases.

In order to alleviate those limitations, we introduce an additional pipeline where we extract a small baseline light field from an MPI and use it to train a continuous scene representation model. As the MPI's geometric errors and uncertainties manifest in the extract light field, we introduce additional geometric constraints using pseudo depth maps. We acquire pseudo-depth maps using pretrained monocular depth estimation model from DPT [97]. Figure 6.4 shows an example light field and corresponding depth maps.

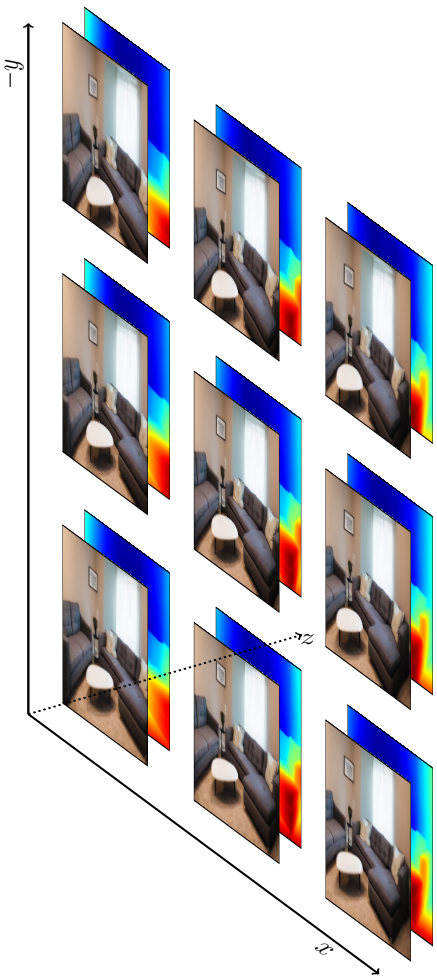


Figure 6.4: **Sample light field and depth maps**: for training implicit scene representation models.

Using the RGB-D light fields we train implicit scene representation model such as InstantNGP [85] and NeRF [83]. As shown in the results section 6.4, an InstantNGP model trained on the MPI light field, shows high quality rendering in both appearance and geometry/depth. Figure 6.6 shows sample InstantNGP renderings and their depth maps at novel camera poses.

We train the downstream InstantNGP model with a combination of photometric and a pseudo depth losses:

$$\mathcal{L}_{i_ngp} = \mathcal{L}_{photo} + \mathcal{L}_{ssi}$$

where $\mathcal{L}_{photo}$ is the $L1$ distance between reconstructed and ground-truth pixel colors, and $\mathcal{L}_{ssi}$ is a scale and shift invariant loss [98] computed between disparity encoded in the instant NGP model and disparity derived from a pre-trained DPT models [97].

6.3.5 Network Architectures

Latent MPI Encoder: first maps each input image (with dimensions $h \times w \times 3$) to a latent representation with dimensions $\frac{h}{k} \times \frac{w}{k} \times z$, using a pre-trained image encoder from [101], where $k = 8$ and $z = 4$. A plane sweep volume built from latent images is then used by a UNet model to predict a latentMPI representation. We use the same network architecture as used in [53]. The UNet model takes input plane-sweep volumes with dimensions $\frac{h}{k} \times \frac{w}{k} \times z \times (n+1) \times d$, where n is number of input views and d is the number of planes in the MPI. In our experiments we use $n = 2$. The model predicts mean and log-variance parameters of a latentMPI representation $\frac{h}{k} \times \frac{w}{k} \times f \times 2$, where $f = 64$. Unlike [53], the encoder and decoder blocks of our UNet model require no time step conditioning. We also apply linear self-attention after each level of the encoder and decoder. During training h and w are set to 384 and 512, respectively.

Latent MPI Decoder: The latentMPI decoder takes an input latentMPI sample of dimensions $\frac{h}{k} \times \frac{w}{k} \times f$ and predicts a standard MPI at full resolution $h \times w \times d \times 4$. We follow the decoder model architecture used in [101]. However, in [101], the decoder is trained to upsample latent representations from $\frac{h}{k} \times \frac{w}{k} \times 4$ to $h \times w \times 3$. Therefore, we change the number of channels in the first and last layers. For computational reasons, we also reduce the number of channels in the model by 2. This model is initialized with random weight parameters and is trained together with the latentMPI encoder model discussed above.

Latent MPI Diffusion Model: learns to denoise latentMPI samples with dimensions $\frac{h}{k} \times \frac{w}{k} \times f$. This model follows the denoising diffusion model architecture in LDM [101]. Except the first and last layers, all layers are initialized using pretrained weights from [101]. The model is further trained as an ϵ parameterised denoising diffusion model as explained above in Section6.3.2.

6.4 Experiments

Dataset. We train and evaluate our proposed Text2MPI approach on RealEstate10K [149], which is a posed multi-view dataset collected from YouTube videos that show multiple rooms of real estate properties. Since RealEstate10K does not provide textual descriptions of the scenes, we use the publicly available BLIP [75] model to compute these automatically.

We sample 180K multi-view training examples, each with four views, from $\approx$ 5600 RealEstate10K clips to train our models, and a further 1500 test examples from a distinct

Figure 6.5: **Visual comparison between methods.** The four columns show results from our Text2MPI model, and the three baseline methods: MiNE [74], AdaMPI [45], and DFM [127]. Our method succeeds in keeping the novel views sharp, in contrast to the rest. The input text prompts for the three rows are: *"a living room with a couch, bookshelf and windows"*, *"a bedroom with a bed, dresser and lamp"*, *"a dining room with a large table and chairs"*.

set of scenes to evaluate them. Each multi-view example consists of three input views, one of which is designated as a reference view, and one target view. We run BLIP [75] on the reference view to derive a text description for the multi-view sample. We train both the latent MPI encoder/decoder (Section 6.3.1), as well as the latent MPI diffusion model (Section 6.3.2) on the same 180K multi-view examples.

Comparison to LDM [101] + single image NVS. We compare our method against three baselines, derived from state-of-the-art single image Novel View Synthesis methods: MiNE [74], AdaMPI [45] and DFM [127]. Since these methods expect an image as their input, we generate the image from the text description either with a pre-trained text-to-image model [101], for quantitative evaluation, or with our Text2MPI method, when we compare results visually.

Quantitative evaluation. We perform quantitative evaluation using the FID [51] and KID [9] measures. We ask all methods to generate a view at a test camera. For a fair comparison, we provide DFM [127] with the test camera pose, in addition to the generated reference image.

In Table 6.1, we measure the realism of the generated views, by computing FID [51]

and KID [9] against real target images from test multi-view samples. In Table 6.2, we measure the effect of Novel View Synthesis on image quality. To this end, we compute FID and KID between generated novel views and images generated by the same pre-trained latent diffusion model [101] that all methods use (ours indirectly, the others directly, as input reference view). The two distances measure quality degradation in this case. In both scenarios, Text2MPI achieves considerably lower FID and KID scores, compared to all other methods. This shows that Text2MPI generates results with both higher realism and better consistency.

	FID ↓	KID ↓
Text2MPI	**45.0972**	**0.0091**
LDM [101]+MiNE [74]	50.2225	0.0128
LDM [101]+AdaMPI [45]	48.5572	0.0109
LDM [101]+DFM [127]	68.5349	0.0295

Table 6.1: **Realism of generated views.** We report FID [51] and KID [9] between generated images at the test camera and real images. Text2MPI generates images with considerably higher level of realism, compared to state-of-the-art single-view NVS techniques.

	FID ↓	KID ↓
Ours	**26.0067**	**0.0037**
LDM [101]+MiNE [74]	35.3818	0.0109
LDM [101]+AdaMPI [45]	32.4770	0.0076
LDM [101]+DFM [127]	67.9246	0.0353

Table 6.2: **Effect of NVS on image quality.** We report FID [51] and KID [9] between images generated at the test camera and images generated by a latent diffusion model [101]. Since all methods depend on it for the reference view, this measures quality degradation from performing the NVS task. Our Text2MPI generates novel views with considerably lower degradation.

Qualitative results. Figure 6.1 presents sample scenes generated with Text2MPI. Figure 6.5, compares our method to the three single-image NVS methods, on a view generated at the test camera. To make outputs visually comparable, we provide the three methods with an image generated by Text2MPI at the reference camera. Our method produces sharp, visually appealing, realistic, and high-quality views of a complex scenes. It exposes minimal artifacts, unlike the other three methods.

Training Implicit Scene Representation Models. As discussed in Section 6.3.4, we robustly renders scenes with higher quality geometry and extended camera motion, by

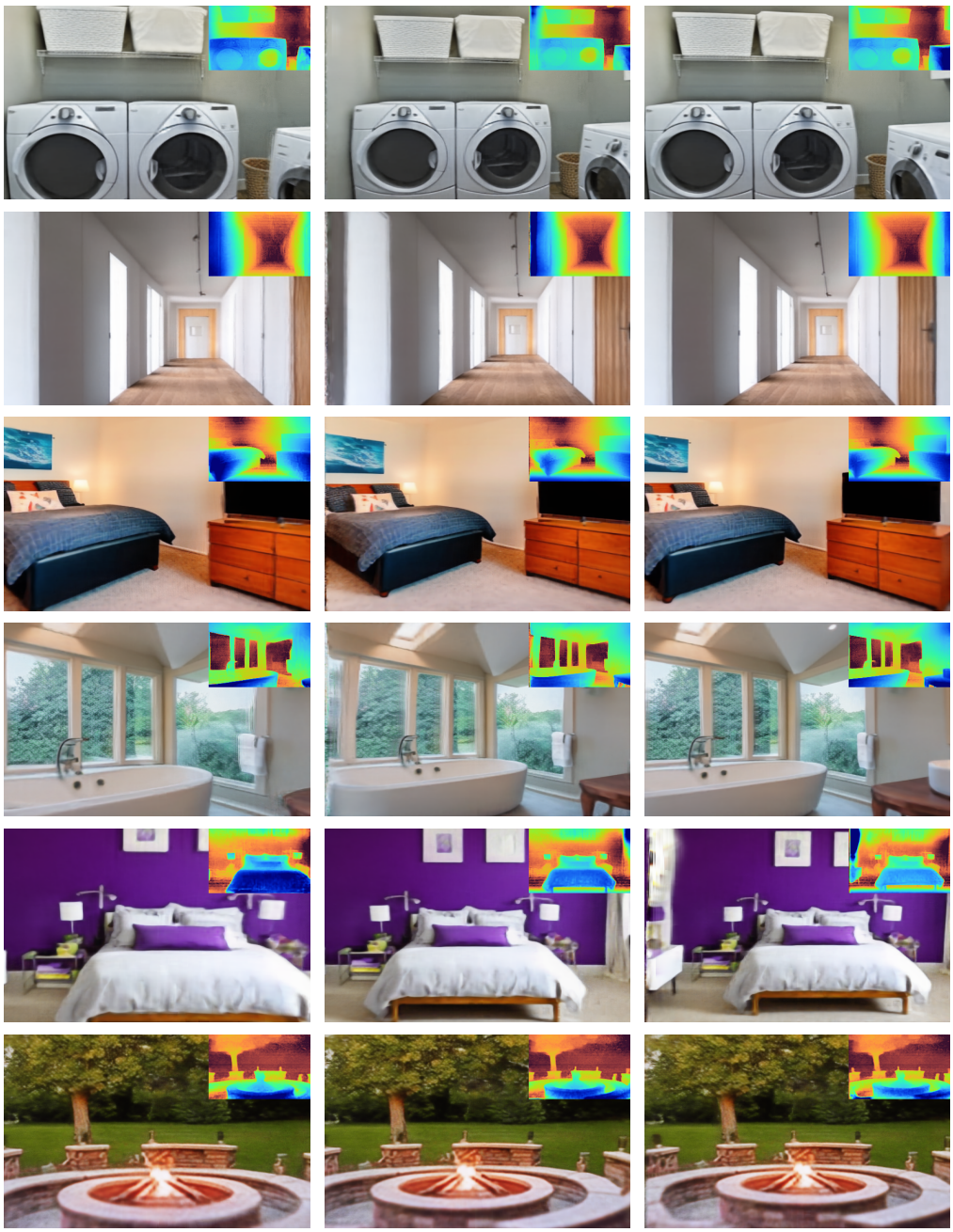


Figure 6.6: **Views rendered with InstantNGP [85]** that is learned from the outputs of Text2MPI. The output of our model contains high quality geometry, as shown in the depth maps extracted from InstantNGP (top right corner). This also allows larger camera translations.

generating a light field from the output of Text2MPI and by training an InstantNGP [85] model on it.

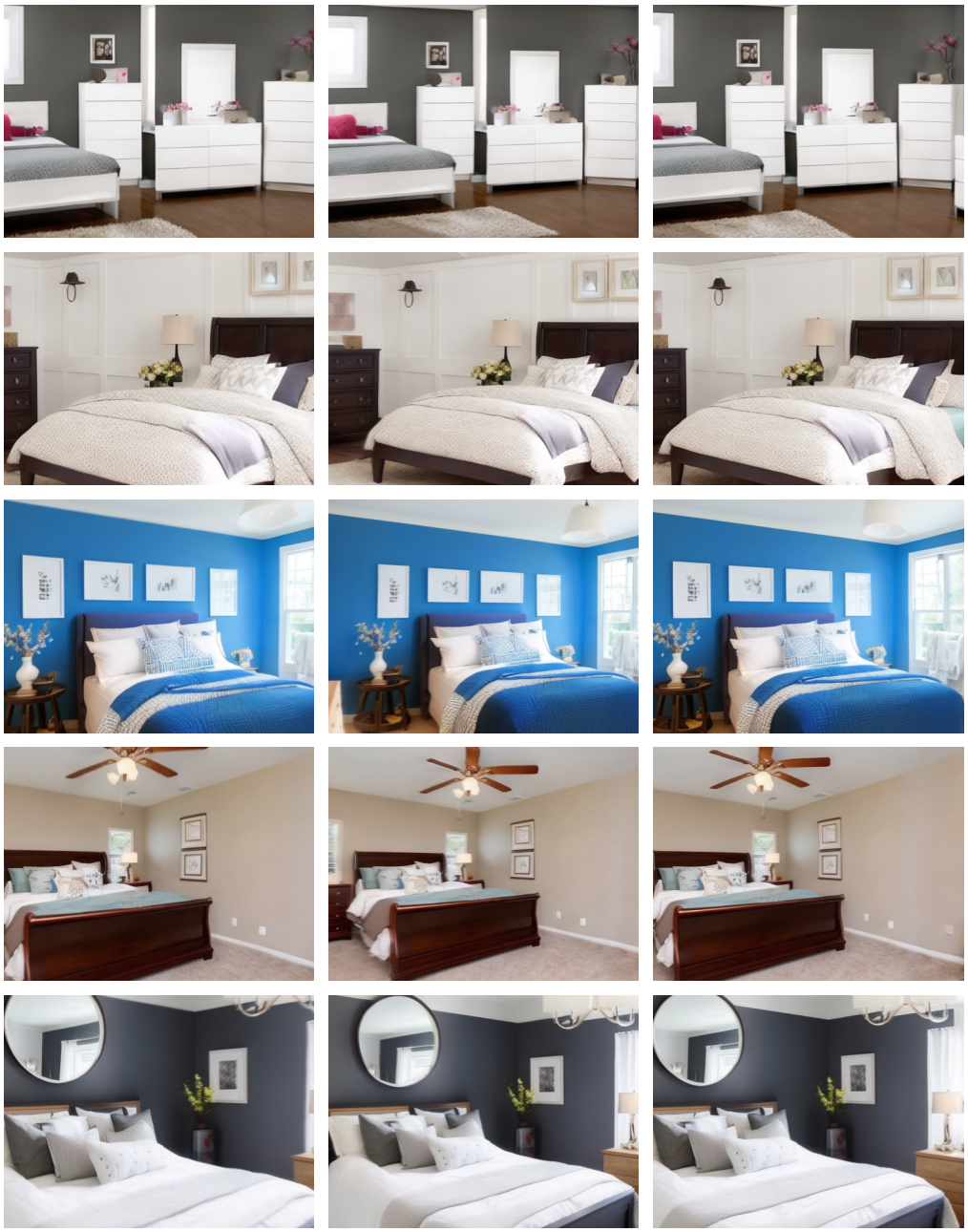


Figure 6.7: **Sample Generated Bedroom Scenes.** We show a small collection of Bedrooms generated with our Text2MPI model. Each row presents 3 views of a single scene, captured from 3 different camera locations and orientations. The scenes are generated using captions taken from test split of the RealEstate10K dataset. For example the scene in row-1 uses the input text: *a bedroom with a bed, dresser and mirror*

We show this in Figure 6.6, same novel view from an InstantNGP model demonstrating higher quality depth maps and more reliable novel views at larger camera motion magnitudes. See the depth maps at the top right corners of each novel view, dark red means further from camera, dark blue means closer to the camera.

Additional visual results. In Figure 6.7, we show sample bedrooms generated using text input captions taken from the RealEsate10K testset. Each scene is rendered from 3 different camera poses. The model is capable of generating scenes that match the desired properties as expressed in the input text captions. For instance the input text for row-1 is the following: *"a bedroom with a bed, dresser and mirror"*, as can be seen in the Figure the scene is a bedroom with a bed, dresser and a bedroom. In row-5, the model takes the following input: *"a bedroom with a bed, dresser and mirror"* and it produces a scene with those objects. Note that the model produces a mirror, which contains a realistic reflection of what might lie behind the camera i.e. a gray wall and white ceiling.

The model is also capable of filling-in scene plausible object properties when no description is given. For example, the third row has an input text *"a bedroom with blue walls and hardwood floors"*, which specifies the color of the walls to be Blue, and leaves out the details on the appearance of the ceiling and bed. Our model predicts a plausible color for the bed the ceiling, by making the bed sheets blue and the ceiling white. This is an indication of our model's understanding of the visual world and also people's taste in modern interior design.

The proposed model is not limited to generating BedRooms, in fact it can generate any scene that is semantically close enough to the real-estate dataset it was trained on. In Figure 6.8, we show sample scenes of houses and living rooms generated with our model. The model can render scenes that are diverse in terms of object categories, scene structure and appearance.

Prompt generalization. We have trained our model with a fairly narrow diversity of images and text prompts, mostly related to real-estate properties. Nevertheless, our model generalizes well due to the generated 2D prior (Section 6.3.3) and it can generate scenes with characteristics that are not present in the training dataset.

We demonstrate this in Figure 6.9, where we show new object types, and scenes with non-photo realistic properties, *e.g.* in the style of *Starry Night* by Van Gogh. The model is trained on a dataset of real world Real-estate scenes, where such artistic concepts are not available. However, by leveraging scene priors from a 2D latent diffusion model trained on a significantly larger and more diverse dataset enables our model to perform inference on out-of-dataset concepts. In the last row Figure 6.9, we also ask our model to render a bed room as it would be built in the medieval period. In addition to global scene properties such as the *starry night* style by van Gogh or a medieval bedroom, we also test the models capability in rendering novel object categories such as dogs. In the second row, of Figure 6.9, we add a dog to the scene.

We show more results on scene with visual concepts that are not present in the scene in Figure 6.10. The caption used to generate each scene is given below the images.

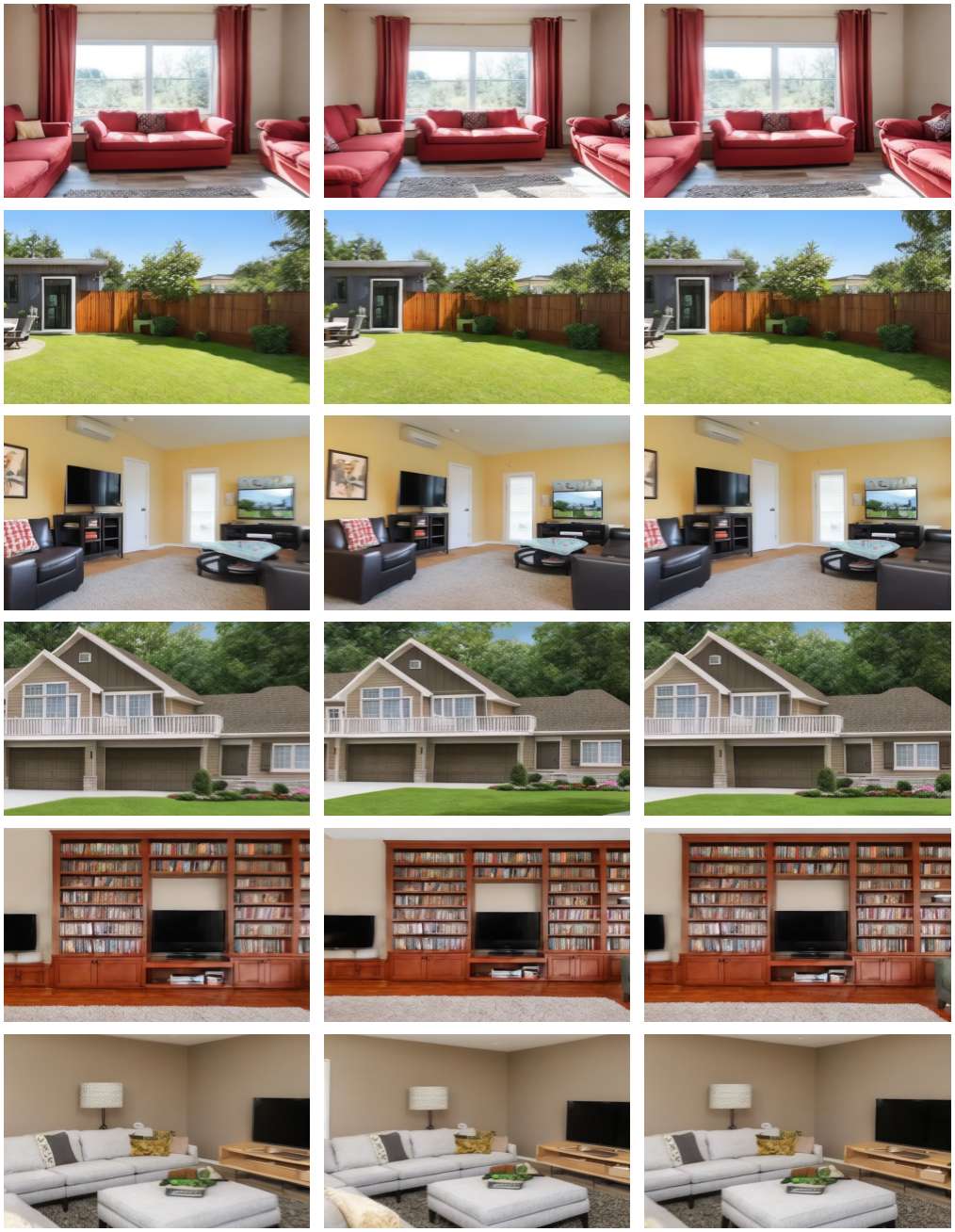


Figure 6.8: **Sample Generated Living Rooms and House Scenes:** Similar to Figure 6.7, each row presents 3 views of a single scene, captured from 3 different camera locations and orientations.

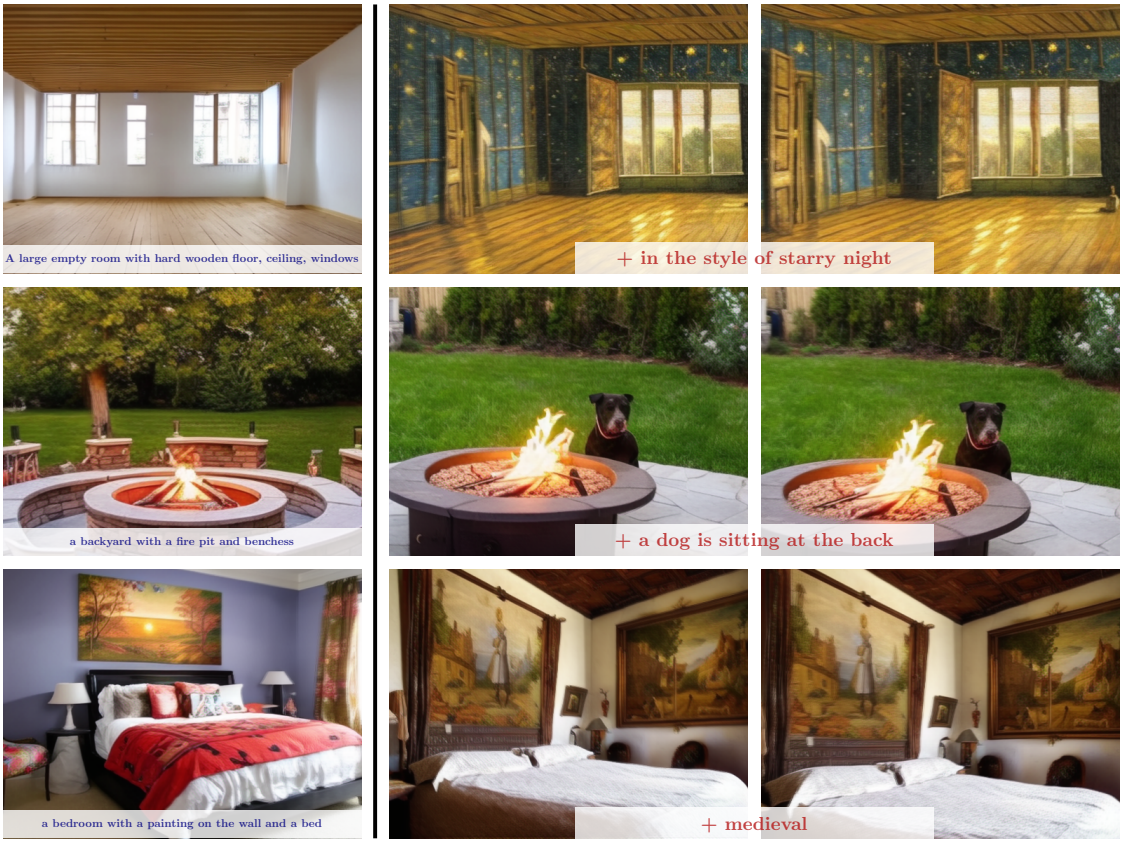


Figure 6.9: **Prompt generalization.** Even though our model was trained on prompts with narrow diversity, it still generalizes to objects and concepts not seen in the training set, due to the generated 2D prior (Section 6.3.3). Column 1 shows a text prompt and a reference view generated for it. Columns 2 and 3 show two views from MPIs generated by our model, when the red text to right is appended to the corresponding prompt to the left.

6.5 Conclusion

In this chapter we presented *Text2MPI*, a latent diffusion model for generating renderable Multiplane Images (MPI) [149] representations of complex scenes, from given textual descriptions. We introduced a compact latent representation that reduces MPI's memory and computation requirements by a factor of 256 and makes training of diffusion models feasible in MPI space.

We proposed a mechanism for transferring the vast knowledge of text-to-image models onto MPI diffusion models. And we showed how to train an InstantNGP model [85] model from generated MPIs to enable larger camera movements. In our experiments, we observed that our method generates 3D representations with complexity matching the scenes in the RealEstate10K [149] dataset and views rendered from our 3D representation were sharp,

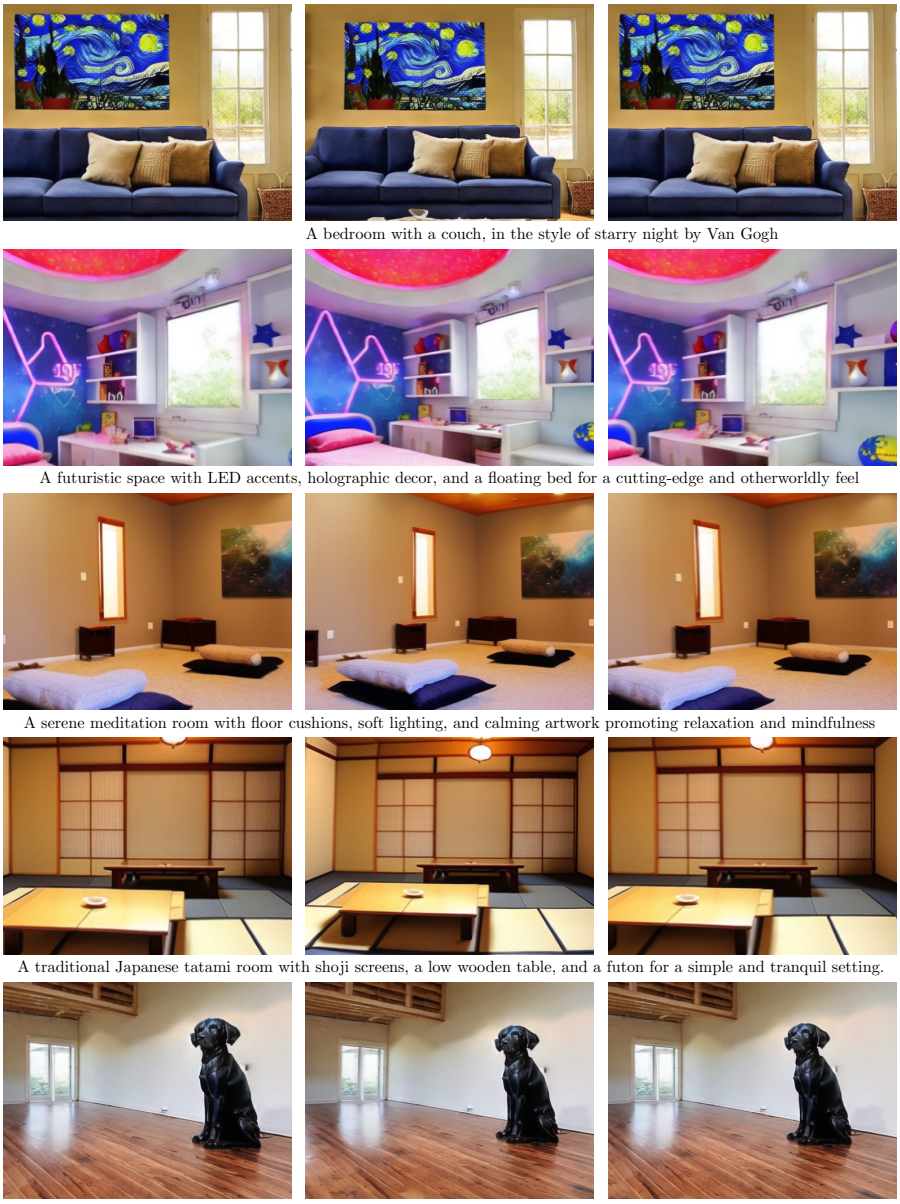


Figure 6.10: **Novel Concepts.** Views rendered with concepts that are not present in the training dataset

visually appealing, realistic, and consistent among each other. We observed generalization on objects classes and concepts that are not present in the training set.

7 Conclusion

This chapter presents a summary of contributions, conclusions, and potential research directions for future work.

Contents

7.1 Thesis Summary **122**
7.2 Future Work **123**

7.1 Thesis Summary

In summary, this thesis presents Novel View Synthesis techniques with progressively simpler input complexity and a wider range of user control. The proposed methods are summarised below.

- FastNVS. In Chapter 3, we introduce the FastNVS technique that achieves new state-of-the-art results for the large baseline view synthesis problem, under multi-view stereo input settings. The proposed method solves the view synthesis problem by decomposing it into proxy geometry estimation and texture inpainting tasks, which are then solved as structured prediction problems using convolutional networks.

- SOMSI. Multi-view stereo-based approaches are not suitable for 360° view synthesis, due to their convolutional architectures that lack invariance to spherical distortions, and cumbersome capturing procedures. In chapter 4, we propose an efficient spherical view synthesis approach for multi-view spherical images, captured with consumer-grade cameras. Our method introduces a novel scene representation based on Multisphere Images (MSI), called Soft Occlusion Multisphere Images (SOMSI). SOMSI effectively models high-dimensional appearance features in MSI representations using a few soft occlusion layers instead of MSI spheres. The proposed method retains the fast rendering of standard MSI representation while producing high quality view synthesis comparable with NeRF-based techniques.

- GVSNet. Chapter 5 introduces the Generative View Synthesis Network (GVSNet) that renders photorealistic novel view images from only a single 2D semantic map. The proposed technique greatly simplifies the complexity of inputs required for view synthesis applications. GVSNet outperforms strong baselines through carefully designed architectures that leverage the structural information present in the input semantics. GVSNet renders geometrically consistent novel view images, while preserving the structures in the input semantics. Moreover, the proposed approach allows users to manipulate the geometry of the scene through edits performed on the input 2D semantic map. Additionally, the scene's appearance can also be changed by swapping the style guidance image with another one.

- Text2MPI. In chapter 6 we present, Text2MPI, a generative model that maps input text descriptions into renderable Multiplane Images (MPI) representations. Text2MPI leverages a compact latent MPI representation, making training large diffusion models in the MPI space possible. The proposed method generates multi-view-consistent photorealistic novel images matching the input textual description. Text2MPI is also able to leverage scene priors learned by 2D diffusion models trained on large collections of image-text pairs. This enables our model to perform view synthesis with objects and visual concepts that are not present in the multi-view training dataset.

In this thesis we introduced models that improved view synthesis in terms of accuracy, speed, efficiency and usability. The key insights behind our methods are the following:

Semantic Cues: We found semantic cues to be crucial sources of geometric information for view synthesis. In GVSNet, we observed that baseline approaches that drop the semantic information early in their pipeline struggle to accurately reconstruct thin structures.

Learned Scene Priors: View synthesis is an ill-posed problem. The problem is further complicated when the number of inputs is reduced, or in the extreme cases when there are no input images. In such difficult scenarios, scene priors from generative models play crucial roles by compensating for lack of observations from which 3D geometry can be inferred. Moreover, generative models enable simpler tools for interaction between the user and the underlying 3D. In our work, we observed that Generative Adversarial Networks and Diffusion Models play crucial roles in creating semantics or text based interfaces with which the user can access the 3D scene geometry and appearance.

Leveraging Geometric Priors: In GVSNet and SOMSI, we achieve efficient scene representation and learning, by leveraging the fact that most of the 3D space is empty and rays in the 3D space intersect with only a few surfaces. In contrast to traditional MPI- and MSI-based approaches, our method assigns appearance features only to a few soft occlusion layers, thereby significantly reducing the scene representation memory footprint.

7.2 Future Work

As part of future work we would like to pursue the quest for more powerful scene representation techniques, that grant users control over the 3D scene. In this thesis, we found text and semantic maps to be effective interfaces between users and scene representations. Therefore, an interesting research direction is the development of scene representation models that are capable of representing complex large scale scenes, while simultaneously allowing user interaction. Developing such models requires moving beyond the current neural scene representation models that treat the 3D world as one big volume, towards models that have semantic understanding of the 3D world. Models with robust semantic scene understanding will be more suitable for seamless interaction between the user and the underlying 3D scenes.

List of Abbreviations

CGI Computer Generated Imagery

CNN Convolutional Neural Network

DDPM Denoising Diffusion Probabilistic Model

DM Diffusion Model

GAN Generative Adversarial Network

GVS Generative View Synthesis

IBR Image-based Rendering

LDM Latent Diffusion Model

MLP Multi-layer Perceptron

MPI Multiplane Images

MSI Multisphere Images

NeRF Neural Radiance Fields

NVS Novel View Synthesis

PSNR Peak Signal-to-Noise Ratio

SfM Structure-from-Motion

SIS Semantic Image Synthesis

SNVS Spherical Novel View Synthesis

SOMSI Soft Occlusion Multisphere Images

SSIM Structural Similarity Index Measure

VAE Variational Autoencoder

VR Virtual Reality

List of Figures

1.1 Image-based Rendering. 13
1.2 Scene Generation, Rendering and Editing using Semantic Maps. 17
1.3 Novel View Synthesis from Text. 18

2.1 The Semantic Image Synthesis (SIS) Problem. 25
2.2 SPADE Generator Architecture. 26
2.3 SPADE Res-Blocks and the SPADE Layer. 27
2.4 Loss of Semantic Information in SIS Networks. 27
2.5 Forward and Backward Diffusion Processes. 28
2.6 LDM Network Architecture. 30
2.7 MPI Scene Representation. 32
2.8 Learning Multiplane Image Representations. 33
2.9 MSI Representation and Rendering. 35
2.10 MODS Network Architecture. 36
2.11 NeRF Scene Representation and Rendering Pipeline. 37
2.12 Effect of Positional Encodings in NeRF. 38

3.1 Sample renderings from the KITTI dataset [39] 42
3.2 Overview of FastNVS approach. 46
3.3 Depth prediction network architecture. 47
3.4 View-warping with forward mapping. 50
3.5 Visual comparison against DeepStereo [36]. 53
3.6 Novel views from $80\,cm$ baseline inputs 56
3.7 Novel views from $160\,cm$ baseline inputs 56

4.1 High quality view synthesis with SOMSI. 61
4.2 Approach Overview. 65
4.3 SOMSI Rendering. 66
4.4 Sample visual results. 70
4.5 Sample visual results, Coffee Area: 70
4.6 Sample visual results, Sea Port: 71
4.7 Sample visual results Coffee Area: 71
4.8 Sample Vanilla-MSI Result. 75

5.1 Generative View Synthesis 81
5.2 GVSNet Overview. 85
5.3 Lifted Semantics. 86

5.4 Sample Visual Results 88
5.5 Consistency across views. 90
5.6 Semantic Editing. 91
5.7 Depth Estimation 92
5.8 Depth Estimation Results. 93

6.1 Text2MPI 103
6.2 An overview of the latent MPI model. 106
6.3 Generated 2D Prior. 109
6.4 Sample light field and depth maps 110
6.5 Visual comparison between methods. 112
6.6 Views rendered with InstantNGP [85] 114
6.7 Sample Generated Bedroom Scenes. 115
6.8 Sample Generated Living Rooms and House Scenes: 117
6.9 Prompt generalization. 118
6.10 Novel Concepts. 119

List of Tables

3.1 Details of the feature extraction stage. 47
3.2 Feature volume filtering network. 48
3.3 Texture inpainting network architecture. 52
3.4 Quantitative evaluation of our method against *DeepStereo*. 54
3.5 Testing different choices of texture inpainting networks. 55

4.1 Comparison with state-of-the-art. 72
4.2 Ablation of appearance feature dimensionality f. 75
4.3 Ablation of occlusion levels k. 75
4.4 The effect of number of reflectance coefficients e for view dependent effects. 76
4.5 SOMSI Network architecture. 77

5.1 Comparisons to Baselines. 89
5.2 Ablative Studies on GVSNet. 89
5.3 Multiview Consistency 90
5.4 Depth accuracy evaluation on CARLA [30] dataset. 92
5.5 Comparing against an SPADE [88]+SM [149] 94
5.6 Semantic evaluations on CARLA [30] dataset. 94
5.7 Semantic Uplifting Network Architecture. 97
5.8 Appearance Decoder Network Architecture. 98

6.1 Realism of generated views. 113
6.2 Effect of NVS on image quality. 113

Bibliography

[1] A Pigment of Your Imagination: GauGAN AI Art Tool Receives 'Best of Show,' 'Audience Choice' Awards at SIGGRAPH, 2019 (Last accessed on 06/01/2020). https://blogs.nvidia.com/blog/2019/07/30/gaugan-ai-painting/. *82*

[2] Edward H Adelson, James R Bergen, et al. The plenoptic function and the elements of early vision. 1991. *42, 44*

[3] Tewodros Amberbir Habtegebrial, Varun Jampani, Orazio Gallo, and Didier Stricker. Generative view synthesis: From single-view semantics to novel-view images. *Advances in Neural Information Processing Systems*, 33:4745–4755, 2020. *15, 61, 64, 74, 105*

[4] Benjamin Attal, Selena Ling, Aaron Gokaslan, Christian Richardt, and James Tompkin. Matryodshka: Real-time 6dof video view synthesis using multi-sphere images. In *European Conference on Computer Vision (ECCV)*, pages 441–459, 2020. *15, 16, 19, 23, 33, 35, 36, 60, 61, 62, 63, 68, 70, 71, 72, 73, 74*

[5] Christian Bailer, Tewodros A Habtegebrial, Kiran Varanasi, and Didier Stricker. Fast dense feature extraction with cnns that have pooling or striding layers. *47*

[6] Christian Bailer, Kiran Varanasi, and Didier Stricker. Cnn-based patch matching for optical flow with thresholded hinge embedding loss. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, page 5, 2017. *47*

[7] Caroline Baillard and Andrew Zisserman. A plane-sweep strategy for the 3d reconstruction of buildings from multiple images. *International Archives of Photogrammetry and Remote Sensing*, 33(B2; PART 2):56–62, 2000. *106*

[8] Tobias Bertel, Neill DF Campbell, and Christian Richardt. Megaparallax: Casual 360Â° panoramas with motion parallax. *IEEE transactions on visualization and computer graphics*, 25(5):1828–1835, 2019. *63*

[9] Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. *arXiv preprint arXiv:1801.01401*, 2018. *112, 113*

[10] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T Barron, Ce Liu, and Hendrik Lensch. Nerd: Neural reflectance decomposition from image collections. In *IEEE*

Conference on Computer Vision and Pattern Recognition (CVPR), pages 12684–12694, 2021. *60*

[11] Michael Broxton, John Flynn, Ryan Overbeck, Daniel Erickson, Peter Hedman, Matthew Duvall, Jason Dourgarian, Jay Busch, Matt Whalen, and Paul Debevec. Immersive light field video with a layered mesh representation. *ACM Transactions on Graphics*, 39(4):86–1, 2020. *33, 60, 63*

[12] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *ACM SIGGRAPH*, 2001. *12, 62, 82, 104*

[13] Yohann Cabon, Naila Murray, and Martin Humenberger. Virtual KITTI 2. In *arXiv Preprint*, 2020. *81, 87, 89, 90, 96*

[14] Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16123–16133, 2022. *102*

[15] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. *43*

[16] Gaurav Chaurasia, Sylvain Duchêne, Olga Sorkine-Hornung, and George Drettakis. Depth synthesis and local warps for plausible image-based navigation. In *ACM Transactions on Graphics*, 2013. *12, 14, 43, 44, 62, 82, 104*

[17] Gaurav Chaurasia, Olga Sorkine, and George Drettakis. Silhouette-aware warping for image-based rendering. In *Eurographics Symposium on Rendering*, 2011. *14, 62, 82*

[18] 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. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14124–14133, 2021. *13, 105*

[19] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *European Conference on Computer Vision (ECCV)*, 2018. *88*

[20] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. *44, 82*

[21] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 279–288. ACM, 1993. *12, 15, 42, 44*

[22] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *ACM SIGGRAPH*, 1993. *62, 82, 104*

[23] Tao Chen, Ming-Ming Cheng, Ping Tan, Ariel Shamir, and Shi-Min Hu. Sketch2photo: Internet image montage. In *ACM Transactions on Graphics (SIGGRAPH ASIA)*, 2009. *82*

[24] Xu Chen, Jie Song, and Otmar Hilliges. Monocular neural image based rendering with continuous view control. In *IEEE International Conference on Computer Vision (ICCV)*, 2019. *14, 80, 82, 83, 88, 89, 94*

[25] Zhaoxi Chen, Guangcong Wang, and Ziwei Liu. Text2light: Zero-shot text-driven hdr panorama generation. *ACM Transactions on Graphics (TOG)*, 41(6):1–16, 2022. *105*

[26] Inchang Choi, Orazio Gallo, Alejandro Troccoli, Min H Kim, and Jan Kautz. Extreme view synthesis. In *IEEE International Conference on Computer Vision (ICCV)*, 2019. *82*

[27] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. *63*

[28] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes dataset for semantic urban scene understanding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. *81, 87, 89, 95, 96*

[29] Paul E Debevec, Camillo J Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. In *ACM SIGGRAPH*, 1996. *12, 82*

[30] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *arXiv Preprint*, 2017. *81, 87, 89, 90, 92, 94, 95, 96, 129*

[31] Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. Learning to generate chairs with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1538–1546, 2015. *43, 44*

[32] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014. *45*

[33] Martin Eisemann, Bert De Decker, Marcus Magnor, Philippe Bekaert, Edilson De Aguiar, Naveed Ahmed, Christian Theobalt, and Anita Sellent. Floating textures. In *Computer graphics forum*, volume 27, pages 409–418. Wiley Online Library, 2008. *62*

[34] Andrew Fitzgibbon, Yonatan Wexler, and Andrew Zisserman. Image-based rendering using image-based priors. *International Journal of Computer Vision*, 63(2):141–151, 2005. *44*

[35] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. *14, 62, 63, 82*

[36] John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. Deepstereo: Learning to predict new views from the world's imagery. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5515–5524, 2016. *13, 14, 15, 19, 43, 44, 45, 52, 53, 54, 55, 56, 62, 104, 127*

[37] David Gadot and Lior Wolf. Patchbatch: a batch augmented loss for optical flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4236–4245, 2016. *47*

[38] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. *arXiv Preprint*, 2021. *60, 61*

[39] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3354–3361. IEEE, 2012. *41, 42, 43, 52, 54, 127*

[40] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. *45, 49, 91, 92, 93*

[41] Michael Goesele, Jens Ackermann, Simon Fuhrmann, Carsten Haubold, Ronny Klowsky, Drew Steedly, and Richard Szeliski. Ambient point clouds for view interpolation. In *ACM SIGGRAPH 2010 papers*, pages 1–6. 2010. *43, 62*

[42] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020. *16, 23, 24, 105*

[43] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. The lumigraph. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 43–54, 1996. *12, 14, 62, 104*

[44] John K Haas. A history of the unity game engine. 2014. *63, 72*

[45] Yuxuan Han, Ruicheng Wang, and Jiaolong Yang. Single-view view synthesis in the wild with learned adaptive multiplane images. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–8, 2022. *112, 113*

[46] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. *31*

[47] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. *51*

[48] Peter Hedman, Tobias Ritschel, George Drettakis, and Gabriel Brostow. Scalable inside-out image-based rendering. *ACM Transactions on Graphics (TOG)*, 35(6):1–11, 2016. *62*

[49] Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. *arXiv Preprint*, 2021. *60, 61*

[50] Aaron Hertzmann, Charles E Jacobs, Nuria Oliver, Brian Curless, and David H Salesin. Image analogies. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pages 557–570. 2023. *105*

[51] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems (NIPS)*, 2017. *88, 89, 112, 113*

[52] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022. *102, 103, 105, 108*

[53] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. *16, 23, 28, 103, 105, 107, 108, 111*

[54] Lukas Höllein, Ang Cao, Andrew Owens, Justin Johnson, and Matthias Nießner. Text2room: Extracting textured 3d meshes from 2d text-to-image models. *arXiv preprint arXiv:2303.11989*, 2023. *105*

[55] Asmaa Hosni, Michael Bleyer, Christoph Rhemann, Margrit Gelautz, and Carsten Rother. Real-time local stereo matching using guided image filtering. In *Multimedia and Expo (ICME), 2011 IEEE International Conference on*, pages 1–6. IEEE, 2011. *44*

[56] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *CoRR*, abs/2106.09685, 2021. *108*

[57] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, volume 1, page 3, 2017. *51, 52*

[58] Hsin-Ping Huang, Hung-Yu Tseng, Hsin-Ying Lee, and Jia-Bin Huang. Semantic view synthesis. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XII 16*, pages 592–608. Springer, 2020. *82*

[59] Hsin-Ping Huang, Hung-Yu Tseng, Hsin-Ying Lee, and Jia-Bin Huang. Semantic view synthesis. In *European Conference on Computer Vision (ECCV)*, 2020. *105*

[60] Sunghoon Im, Hae-Gon Jeon, Stephen Lin, and In So Kweon. DpsNet: End-to-end deep plane sweep stereo. In *arXiv Preprint*, 2019. *88, 95, 96*

[61] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*, 2016. *44*

[62] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. *24, 80, 82, 105*

[63] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 28:2017–2025, 2015. *32, 49, 50*

[64] Matthew Johnson, Gabriel J Brostow, Jamie Shotton, Ognjen Arandjelovic, Vivek Kwatra, and Roberto Cipolla. Semantic photo synthesis. In *Computer Graphics Forum (CGF)*, 2006. *82*

[65] Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. Learning-based view synthesis for light field cameras. *ACM Transactions on Graphics (TOG)*, 35(6):1–10, 2016. *44*

[66] Animesh Karnewar, Andrea Vedaldi, David Novotny, and Niloy J Mitra. Holodiffusion: Training a 3d diffusion model using 2d images. In *Proceedings of the*

IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 18423–18433, 2023. *102, 105*

[67] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-end learning of geometry and context for deep stereo regression. *arXiv preprint arXiv:1703.04309*, 2017. *45, 48*

[68] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023. *12*

[69] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *arXiv Preprint*, 2014. *54, 77, 87, 96*

[70] Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. *Advances in neural information processing systems*, 28, 2015. *26, 106*

[71] Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems*, pages 2539–2547, 2015. *43*

[72] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 239–248. IEEE, 2016. *45*

[73] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 31–42, 1996. *12, 14, 42, 62, 104*

[74] Jiaxin Li, Zijian Feng, Qi She, Henghui Ding, Changhu Wang, and Gim Hee Lee. Mine: Towards continuous depth mpi with nerf for novel view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12578–12588, 2021. *112, 113*

[75] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International Conference on Machine Learning*, pages 12888–12900. PMLR, 2022. *111, 112*

[76] Zhengqi Li, Wenqi Xian, Abe Davis, and Noah Snavely. Crowdsampling the plenoptic function. In *European Conference on Computer Vision (ECCV)*, pages 178–196. Springer, 2020. *15, 61, 64*

[77] Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In *European Conference on Computer Vision (ECCV)*, 2018. *91*

[78] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. *107, 108*

[79] Bicheng Luo, Feng Xu, Christian Richardt, and Jun-Hai Yong. Parallax360: Stereoscopic 360 scene representation for head-motion parallax. *IEEE transactions on Visualization and Computer Graphics*, 24(4):1545–1553, 2018. *63*

[80] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7210–7219, 2021. *13, 60, 69*

[81] Leonard McMillan and Gary Bishop. Plenoptic modeling: An image-based rendering system. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 39–46. ACM, 1995. *42*

[82] Leonard McMillan and Steven Gortler. Image-based rendering: A new interface between computer vision and computer graphics. *ACM Siggraph Computer Graphics*, 33(4):61–64, 1999. *14*

[83] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision (ECCV)*, pages 405–421. Springer, 2020. *12, 13, 14, 16, 18, 19, 23, 31, 36, 37, 38, 60, 61, 62, 69, 70, 71, 72, 73, 74, 104, 110*

[84] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv Preprint*, 2018. *98*

[85] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022. *12, 15, 16, 102, 104, 105, 110, 114, 118, 128*

[86] Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. Hologan: Unsupervised learning of 3d representations from natural images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. *82*

[87] Simon Niklaus, Long Mai, Jimei Yang, and Feng Liu. 3d ken burns effect from a single image. *ACM Transactions on Graphics (ToG)*, 38(6):1–15, 2019. *14*

[88] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. *23, 24, 25, 26, 27, 80, 81, 82, 83, 84, 85, 87, 88, 89, 90, 94, 95, 98, 105, 107, 129*

[89] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NIPS)*, 2019. *77, 87*

[90] Eric Penner and Li Zhang. Soft 3d reconstruction for view synthesis. *ACM Transactions on Graphics (TOG)*, 36(6):235, 2017. *14, 43, 44*

[91] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2016. *34, 64, 67*

[92] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022. *102, 105*

[93] 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*, pages 10318–10327, 2021. *13, 105*

[94] Albert Pumarola, Stefan Popov, Francesc Moreno-Noguer, and Vittorio Ferrari. C-flow: Conditional generative flow models for images and 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. *103, 108*

[95] Amit Raj, Srinivas Kaza, Ben Poole, Michael Niemeyer, Nataniel Ruiz, Ben Mildenhall, Shiran Zada, Kfir Aberman, Michael Rubinstein, Jonathan Barron, et al. Dreambooth3d: Subject-driven text-to-3d generation. *arXiv preprint arXiv:2303.13508*, 2023. *105*

[96] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022. *102, 105*

[97] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12179–12188, 2021. *107, 110, 111*

[98] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE transactions on pattern analysis and machine intelligence*, 44(3):1623–1637, 2020. *107, 111*

[99] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. *arXiv preprint arXiv:2103.13744*, 2021. *60, 61*

[100] Konstantinos Rematas, Andrew Liu, Pratul P Srinivasan, Jonathan T Barron, Andrea Tagliasacchi, Thomas Funkhouser, and Vittorio Ferrari. Urban radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12932–12942, 2022. *105*

[101] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. *23, 24, 28, 30, 102, 103, 105, 106, 107, 108, 111, 112, 113*

[102] 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*. Springer, 2015. *55, 87*

[103] Antoni Rosinol, John J Leonard, and Luca Carlone. Nerf-slam: Real-time dense monocular slam with neural radiance fields. *arXiv preprint arXiv:2210.13641*, 2022. *13, 105*

[104] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. 2022. *108*

[105] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022. *102, 103, 105, 108*

[106] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *IEEE International Conference on Computer Vision (ICCV)*, pages 9339–9347, 2019. *72*

[107] Daniel Scharstein. Stereo vision for view synthesis. In *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR'96, 1996 IEEE Computer Society Conference on*, pages 852–858. IEEE, 1996. *42, 44*

[108] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016. *37, 72, 77*

[109] Steven M Seitz and Charles R Dyer. Physically-valid view synthesis by image interpolation. In *Representation of Visual Scenes, 1995.(In Conjuction with ICCV'95), Proceedings IEEE Workshop on*, pages 18–25. IEEE, 1995. *42, 44*

[110] Steven M Seitz and Charles R Dyer. View morphing. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 21–30. ACM, 1996. *44*

[111] Steven M Seitz and Kiriakos N Kutulakos. Plenoptic image editing. *International Journal of Computer Vision*, 48:115–129, 2002. *14*

[112] Ana Serrano, Incheol Kim, Zhili Chen, Stephen DiVerdi, Diego Gutierrez, Aaron Hertzmann, and Belen Masia. Motion parallax for 360 rgbd video. *IEEE Transactions on Visualization and Computer Graphics*, 2019. *63*

[113] Harry Shum and Sing Bing Kang. Review of image-based rendering techniques. In *Visual Communications and Image Processing 2000*, volume 4067, pages 2–13. SPIE, 2000. *14*

[114] Heung-Yeung Shum, Shing-Chow Chan, and Sing Bing Kang. *Image-based rendering*. Springer Science & Business Media, 2008. *14*

[115] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. *27, 88*

[116] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM siggraph 2006 papers*, pages 835–846. 2006. *13*

[117] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015. *16, 23, 28, 102, 103, 105, 107, 108*

[118] Pratul P Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T Barron. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7495–7504, 2021. *60, 105*

[119] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. *82*

[120] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019. *72*

[121] Richard Szeliski and Polina Golland. Stereo matching with transparency and matting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1998. *13, 60, 63, 80, 82, 83*

[122] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8248–8258, 2022. *105*

[123] Matthew Tancik, Pratul P Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *arXiv preprint arXiv:2006.10739*, 2020. *38*

[124] Shitao Tang, Fuyang Zhang, Jiacheng Chen, Peng Wang, and Yasutaka Furukawa. Mvdiffusion: Enabling holistic multi-view image generation with correspondence-aware diffusion. *arXiv preprint arXiv:2307.01097*, 2023. *105*

[125] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in neural information processing systems*, 30, 2017. *108*

[126] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Multi-view 3d models from single images with a convolutional network. In *European Conference on Computer Vision*, pages 322–337. Springer, 2016. *43, 44*

[127] Ayush Tewari, Tianwei Yin, George Cazenavette, Semon Rezchikov, Joshua B Tenenbaum, Frédo Durand, William T Freeman, and Vincent Sitzmann. Diffusion with forward models: Solving stochastic inverse problems without direct supervision. *arXiv preprint arXiv:2306.11719*, 2023. *112, 113*

[128] Richard Tucker and Noah Snavely. Single-view view synthesis with multiplane images. In *arXiv Preprint*, 2020. *12, 14, 31, 82, 105*

[129] Shubham Tulsiani, Richard Tucker, and Noah Snavely. Layer-structured 3d scene inference via view synthesis. In *European Conference on Computer Vision (ECCV)*, 2018. *82*

[130] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. Demon: Depth and motion network for learning monocular stereo. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. *92*

[131] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T Barron, and Pratul P Srinivasan. Ref-nerf: Structured view-dependent appearance for neural

radiance fields. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5481–5490. IEEE, 2022. *13, 105*

[132] Qianqian Wang, Yen-Yu Chang, Ruojin Cai, Zhengqi Li, Bharath Hariharan, Aleksander Holynski, and Noah Snavely. Tracking everything everywhere all at once. *arXiv preprint arXiv:2306.05422*, 2023. *13, 105*

[133] 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 Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. *24, 27, 44, 82, 105*

[134] 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. *49*

[135] 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. *73*

[136] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. SynSin: End-to-end view synthesis from a single image. In *arXiv Preprint*, 2020. *12, 14, 15, 61, 64, 74, 80, 82, 83*

[137] Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. Nex: Real-time view synthesis with neural basis expansion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8534–8543, 2021. *62, 68*

[138] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. In *Computer Graphics Forum*, volume 41, pages 641–676. Wiley Online Library, 2022. *14*

[139] Jimei Yang, Scott E Reed, Ming-Hsuan Yang, and Honglak Lee. Weakly-supervised disentangling with recurrent transformations for 3d view synthesis. In *Advances in Neural Information Processing Systems*, pages 1099–1107, 2015. *43*

[140] Lin Yen-Chen, Pete Florence, Jonathan T Barron, Tsung-Yi Lin, Alberto Rodriguez, and Phillip Isola. Nerf-supervision: Learning dense object descriptors from neural radiance fields. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6496–6503. IEEE, 2022. *13*

[141] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4578–4587, 2021. *60*

[142] Jure Zbontar and Yann LeCun. Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research*, 17(1-32):2, 2016. *45, 47*

[143] Fang-Lue Zhang, Jue Wang, Eli Shechtman, Zi-Ye Zhou, Jia-Xin Shi, and Shi-Min Hu. Plenopatch: Patch-based plenoptic image manipulation. *IEEE transactions on visualization and computer graphics*, 23(5):1561–1573, 2016. *14*

[144] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3836–3847, October 2023. *108*

[145] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. *88, 107*

[146] Shangzhan Zhang, Sida Peng, Tianrun Chen, Linzhan Mou, Haotong Lin, Kaicheng Yu, Yiyi Liao, and Xiaowei Zhou. Painting 3d nature in 2d: View synthesis of natural scenes from a single semantic mask. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8518–8528, 2023. *105*

[147] Yiran Zhong, Yuchao Dai, and Hongdong Li. Self-supervised learning for stereo matching with self-improving ability. *arXiv preprint arXiv:1709.00930*, 2017. *45*

[148] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, volume 2, page 7, 2017. *45*

[149] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo Magnification: Learning view synthesis using multiplane images. In *ACM Transactions on Graphics (SIGGRAPH)*, 2018. *12, 13, 14, 23, 31, 32, 33, 60, 62, 63, 80, 81, 82, 83, 87, 88, 89, 90, 94, 102, 103, 104, 105, 106, 111, 118, 129*

[150] Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. View synthesis by appearance flow. In *European Conference on Computer Vision (ECCV)*, 2016. *43, 44, 83, 88, 89, 94*

[151] Yi Zhu, Karan Sapra, Fitsum A Reda, Kevin J Shih, Shawn Newsam, Andrew Tao, and Bryan Catanzaro. Improving semantic segmentation via video propagation and label relaxation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. *88, 95*

[152] C Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. High-quality video view interpolation using a layered representation. *ACM Transactions on Graphics*, 2004. *13, 43, 82*

Curriculum Vitae

Tewodros Amberbir Habtegebrial

PhD Candidate · Computer Science

tedyhabtegebrial@gmail.com | https://tedyhabtegebrial.github.io

Education

RPTU Kaiserslautern-Landau — *Kaiserslautern, Germany*
PhD, Computer Science — *Jan. 2017 - Dec. 2023*
- Advisor: Prof. Didier Stricker
- PhD Thesis: ”Learning View Synthesis from Minimal Scene Specifications”.

RPTU Kaiserslautern-Landau — *Kaiserslautern, Germany*
MSc, Computer Science — *Nov. 2014 - Jan. 2017*
- Advisors: Dr. Christian Bailer, Prof. Didier Stricker
- Thesis: ”Novel-view Synthesis with Deep Convolutional Networks”.
- Specialization: Intelligent Systems
- Overall Grade: 1.3

Mekelle Institute of Technology — *Mekelle, Ethiopia*
BSc, Electronics and Communications Engineering — *Oct. 2007 - Aug. 2012*
- Advisor: Dr. Workineh Gebeyehu
- Thesis: ”Visual Object Tracking Using Particle Filtering”.
- Overall Grade: 3.79/4.00

Professional Experience

Machine Learning Engineer
Snap Inc — *Sep. 2024 - present*
Working on Machine Learning and Computer Vision

Researcher — *Kaiserslautern, Germany*
German Research Center for Artificial Intelligence (DFKI) — *Jan. 2017 - Nov. 2023*
Research projects on Image-based Rendering, VR and 3D Reconstruction

Software Engineer — *Addis-Ababa, Ethiopia*
iCog Labs — *Oct. 2013 - Oct. 2014*
I worked on research projects in unsupervised learning from videos and single image collections

Internships and Student Researcher Positions

Google Research — *Remote*
Student Researcher: with Dr. Stefan Popov — *Aug. 2022 - Mar. 2023*
- Topic: “Multi-view Consistent Diffusion Models”

Meta Reality Labs — *Remote*
Intern: with Dr. Chakravarty Chaitanya, Dr. Federico Perazzi — *Jun. - Sep. 2021*
- Topic: “Low-Light NeRF”

Google Summer of Code — *Summer 2015*
with OpenCog Project

Awards & Fellowships

Google PhD Fellowship *2021-2023*
Google Research
Research area: Computer Vision

CS Graduate Scholarship *2015-2021*
RPTU Kaiserslautern-Landau *Kaiserslautern, Germany*
Graduate Scholarship to pursue Master in Computer Science

Merit Scholarship *2008-2012*
Mekelle Institute of Technology *Mekelle, Ethiopia*

Teaching Experience

3D Computer Vision *Kaiserslautern, Germany*
RPTU Kaiserslautern-Landau *Winter Semesters 2018-2022*

Graph Theory *Kaiserslautern, Germany*
RPTU Kaiserslautern-Landau *Summer Semester 2016*

Case-based Reasoning *Kaiserslautern, Germany*
RPTU Kaiserslautern-Landau *Winter Semester 2016*

Programming Languages

Python, Lua, Matlab, C, C++

CUDA, PyCUDA

Selected Publications

TalkingPose: Efficient Face and Gesture Animation with Feedback-guided Diffusion Model. Javanmardi etal. Under Review ICCV 2025

1D-DiffNPHR: 1D Diffusion Neural Parametric Head Reconstruction using a Single Image. Pragati Jaiswal, Tewodros **Habtegebrial**, and, Stricker Didier. ICPRAM, 2025.

Text2MPI: Learning to Generate Multi-Plane Images from Text. Tewodros **Habtegebrial**, Stefan Popov, Didier, Stricker, Varun Jampani and Vittorio Ferrari. Preprint, 2024.

SOMSI: Spherical Novel View Synthesis with Soft Occlusion Multi-sphere Images. Tewodros **Habtegebrial**, Christiano Gava, Marcel Rogge, Stricker Didier and Varun Jampani. CVPR, 2022.

Generative View Synthesis: From Single-view Semantics to Novel-view Images. Tewodros **Habtegebrial**, Varun Jampani, Orazio Gallo, and Didier Stricker. NeuRIPS, 2020.

Fast View Synthesis with Deep Stereo Vision. Tewodros **Habtegebrial**, Kiran Varanasi, Christian Bailer, and Didier Stricker. VISAPP, 2019.

Deep Convolutional Networks For Snapshot Hyperspectral Demosaicking. Tewodros **Habtegebrial**, Gerd Reis, and Didier Stricker. WHISPERS, 2019.

Fast Feature Extraction with CNNs with Pooling Layers. Christian Bailer, Tewodros **Habtegebrial**, Kiran Varanasi, and Didier Stricker. BMVC, 2018.

SphereGlue: Learning Keypoint Matching on High Resolution Spherical Images. Christiano Gava, Vishal Mukunda, Tewodros **Habtegebrial**, Federico Raue, Sebastian Palacio and Andreas Dengel. CVPR-Workshops, 2023

Amharic Text Image Recognition: Database, Algorithm, and Analysis. Birhanu Hailu Belay, Tewodros **Habtegebrial**, Marcus Liwicki, Gebyehu Belay and Didier Stricker. International Conference on Document Analysis and Recognition (ICDAR), 2019

Factored Convolutional Neural Network For Amharic Ccharacter Image Recognition. Birhanu Hailu Belay, Tewodros **Habtegebrial**, Marcus Liwicki, Gebeyehu Belay and Didier Stricker. International Conference on Image Processing (ICIP), 2019