Towards Computationally Efficient, Photorealistic, and Scalable 3D Generative Modelling

Animesh Karnewar

A dissertation submitted in fulfillment of the requirements for the degree of **Doctor of Philosophy**

of

University College London.

Department of Computer Science University College London

July 2, 2024

I, Animesh Karnewar, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

GM (Generative Modelling) is a class of self-supervised Machine Learning which finds applications in synthetic data generation, semantic representation learning, and various creative and artistic fields. GM (aka. Generative AI) seemingly holds the potential for the next breakthrough in AI; of which, the recent successes in LLMs, text-to-image synthesis and text-to-video synthesis serve as formidable testament. The way these generative models have revolutionized the process of 2D content creation, we can expect that 3D generative modelling will also contribute significantly towards simplifying the process of 3D content creation. However, it is non-trivial to extend the 2D generative algorithms to operate on 3D data managing various factors such as the inherent data-sparsity, the growing memory requirements, and the computational complexity. The application of Generative Modelling to 3D data is made even harder due to the pertaining challenges: firstly, finding a large quantity of 3D training data is much more complex than 2D images; and secondly, there is no de-facto representation for 3D assets, where various different representations such as point-clouds, meshes, voxel-grids, neural (MLP)s, etc. are used depending on the application. Thus, with the goal of ultimately enabling 3D Generative Models, and considering the aforementioned challenges, I propose this thesis which makes substantial strides "Towards Computationally Efficient, Photorealistic, and Scalable 3D Generative Modelling".

Impact Statement

This thesis, shines a key new insight into our understanding of 3D representations (chapter 4); proposes first ever approaches for certain tasks that were deemed impossible earlier, such as training a 3D generative model using a single 3D scene (chapter 5), and training a 3D diffusion model only using 2D images (chapter 6); as well as proposes a highly unique and intriguing mathematical framework of generative modelling for handling arbitrary 3D representations that exist today or will be proposed in the future (chapter 8).

Apart from the research contributions, this thesis has tangible real-world impact. For instance, the latest release of the Mitsuba renderer version 3, demonstrates fitting ReLU-Fields (chapter 4) as a showcase example of the framework [1]; the 5 publications in total have received over 150 citations; and, the code repositories of publications have in total received more than 250 stars on GitHub, at the time of writing this thesis within a period of less than 2 years.

All of our novel approaches have been published at the premier venues for scholarly work in the fields of 3D Vision (3DV), Computer Vision (CVPR and ICCV/ECCV) and Computer Graphics (SIGGRAPH). To encourage further research on our approaches, I release the code, datasets and additional results for most of the publication as allowed by the university, funding agency and company policies. For all the publications, I release supplementary explanatory videos and project websites for efficient dissemination of the proposed research ideas.

Acknowledgements

I would like to extend my sincere gratitude to the supervisors Prof. Niloy J. Mitra and Prof. Tobias Ritschel for their invaluable advice and their continuous support towards realising this research agenda. Niloy's advise has been indispensable for keeping the research on track, especially on such a topic which is so prone to divergence. Looking back, it's impossible to imagine even a single publication getting accepted at the top venues, without his motivation, his dedication, and his meticulousness, in spite of the hectic deadlines. He caught the minutest of mistakes, which not only helped the research go forward, but also allowed to present and disseminate the works in an impactful manner. I am also thankful to Tobias for all the technical contributions and for mentoring me in this journey. I have learned almost all my research-illustration skills from him.

It has been a journey in learning and an absolute pleasure working with my research collaborators David Novotny and Prof. Andrea Vedaldi. I learned from David how to approach a goal oriented research in an impactful manner. His discipline of work has helped us push the research directions towards fruitful results efficiently and productively. Not to mention, I have certainly picked up in-numerous coding and implementation best practices from him. Andrea's critical and mathematical thinking has helped in developing the research ideas into their best forms. His tremendous experience and his foresight not only made all our research discussions fruitful, but they also provided big learning lessons to me.

I would like to mention special thanks to Oliver Wang, with whom my association started even before the Doctoral endeavour. My research journey began with the MSG-GAN project which was mentored by Oliver; and the discussions that we used to have inspired me to officially pursue a PhD, quitting my full time job (R&D Engi-

Acknowledgements

neer, TomTom Netherlands). Because I was still collaborating with him in the initial projects during PhD, the transition from "free-time" research to "full-time" research became seamless for me. Most importantly, his guidance regarding incorporating a holistic-view of research, especially in an area as crowded and as competitive as the one I am pursuing, has not only been a life-saver, but a life-long learning that I have imbibed in me which will drive my future research.

More than just extending gratitude, I would like to dedicate this research thesis to my parents, especially to my mom, without whom I would definitely not have been able to complete this research thesis. I would like to thank my family for providing the love, care and support during this arduous, but fulfilling, journey. My sincere homage to my grandfather Dr. C.N. Maggirwar (ex. Director, GSDA India), who has been the source of inspiration and motivation for us. He is my role-model, and I draw immense inspiration from his contributions to the field of Geology, his contributions towards the society, and how he fulfilled all his responsibilities in life. Whatever I have achieved in my life, or ever will, is all because of him.

Huge thanks to the ITN-PRIME, being a part of which has been a tremendous privilege, honour, and a pleasure. Big thanks to my peers, the ESRs (Early Stage Researchers) of the training network https://prime-itn.eu/about/people/fellows/; the discussions and collaborations with whom, allowed for the cross-pollination of ideas from diverse research-sub-fields of Computer Graphics. I thoroughly enjoyed the time spent with y'all during the PRIME activities, the training sessions, and of course, our hangouts. I am also grateful for the opportunity to connect and share my research with so many experts from academia and the industry. Special thanks to Eva Šauerová and Markéta Tomková for looking after all the PRIME organization, administration, and reporting, so timely and particularly.

Lastly, I would like to acknowledge and sincerely thank all the funding agencies without whom, this research would certainly not have been possible. ITN-PRIME (European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 956585), UCL AI centre, gifts from Adobe research, and last but not the least, The Rabin Ezra Scholarship Trust.

Contents

1	Introduction				
	1.1	Why 3	D?	11	
	1.2	What i	s Generative Modelling?	13	
	1.3	What o	can Generative Modelling do?	14	
		1.3.1	Success stories of 2D Generative Modelling	15	
		1.3.2	Possible applications of 3D Generative Models	17	
	1.4	Scope	of the thesis	19	
	1.5	Challe	nges and Opportunities	20	
2	Prel	iminari	es	23	
	2.1	Deep l	Learning	23	
		2.1.1	Neural Networks	25	
		2.1.2	Mathematical Optimization	29	
	2.2 3D Radiance Fields				
		2.2.1	Differentiable volumetric rendering	33	
		2.2.2	End-to-end 3D reconstruction pipeline	36	
		2.2.3	Summary	37	
	2.3	Genera	ative Models	38	
		2.3.1	Generative Adversarial Networks	38	
		2.3.2	Diffusion Models	41	
3	Lite	rature s	survey	45	
3.1 Discrete sample based representations					

Contents

	3.2	Differentiable rendering.	46
	3.3	Learned neural representations	46
	3.4	2D-to-3D Encoding	47
	3.5	2D generative models.	48
	3.6	3D generative models	49
4	ReL	U-Fields: The Little Non-linearity That Could	52
	4.1	Background and Contributions	52
	4.2	Introduction	53
	4.3	It's just a little ReLU	55
	4.4	Applications	57
		4.4.1 Radiance Fields	58
		4.4.2 Occupancy Fields	61
	4.5	Limitations	63
	4.6	Summary	64
5	3inC	GAN: Learning a 3D Generative Model from Images of a Self-similar	
5	3in Scer	GAN: Learning a 3D Generative Model from Images of a Self-similar ne	66
5	3in (Scer 5.1	GAN: Learning a 3D Generative Model from Images of a Self-similar ne Background and Contributions	66 66
5	3in (Scer 5.1 5.2	GAN: Learning a 3D Generative Model from Images of a Self-similar ne Background and Contributions Introduction	66 66 67
5	3in (Scer 5.1 5.2 5.3	GAN: Learning a 3D Generative Model from Images of a Self-similar ne Background and Contributions Introduction 3inGAN approach	66 66 67 70
5	3in (Scer 5.1 5.2 5.3	GAN: Learning a 3D Generative Model from Images of a Self-similar ne Background and Contributions Introduction 3inGAN approach 5.3.1 Representation	66 66 67 70 72
5	3in (Scer 5.1 5.2 5.3	GAN: Learning a 3D Generative Model from Images of a Self-similar ne Background and Contributions Introduction 3inGAN approach 5.3.1 Representation 5.3.2 Generation	66 66 67 70 72 73
5	3in (Scer 5.1 5.2 5.3	GAN: Learning a 3D Generative Model from Images of a Self-similar ne Background and Contributions Introduction 3inGAN approach 5.3.1 Representation 5.3.2 Generation Generation	66 66 67 70 72 73 75
5	3in (Scer 5.1 5.2 5.3 5.4	GAN: Learning a 3D Generative Model from Images of a Self-similar ne Background and Contributions Introduction 3inGAN approach 5.3.1 Representation 5.3.2 Generation Self-similar 5.4.1 Qualitative Evaluation	66 66 67 70 72 73 75 77
5	3in (Scer 5.1 5.2 5.3 5.4	GAN: Learning a 3D Generative Model from Images of a Self-similar ne Background and Contributions Introduction 3inGAN approach 5.3.1 Representation 5.3.2 Generation 5.4.1 Quantitative Evaluation	 66 67 70 72 73 75 77 81
5	3in (Scer 5.1 5.2 5.3 5.4	GAN: Learning a 3D Generative Model from Images of a Self-similar ne Background and Contributions Introduction 3inGAN approach 5.3.1 Representation 5.3.2 Generation 5.4.1 Qualitative Evaluation 5.4.2 Quantitative Evaluation	 66 67 70 72 73 75 77 81 81
5	3in (Scer 5.1 5.2 5.3 5.4 5.4	GAN: Learning a 3D Generative Model from Images of a Self-similar ne Background and Contributions Introduction 3inGAN approach 5.3.1 Representation 5.3.2 Generation 5.4.1 Qualitative Evaluation 5.4.2 Quantitative Evaluation Summary	 66 67 70 72 73 75 77 81 81 82
5	3in (Scer 5.1 5.2 5.3 5.4 5.4 5.5 5.6 Hole	GAN: Learning a 3D Generative Model from Images of a Self-similar ne Background and Contributions Introduction 3inGAN approach 5.3.1 Representation 5.3.2 Generation 5.4.1 Qualitative Evaluation 5.4.2 Quantitative Evaluation Summary Summary	 66 67 70 72 73 75 77 81 81 82 83
5	3in (Scer 5.1 5.2 5.3 5.4 5.4 5.5 5.6 Hold 6.1	GAN: Learning a 3D Generative Model from Images of a Self-similar ne Background and Contributions Introduction 3inGAN approach 5.3.1 Representation 5.3.2 Generation 5.4.1 Qualitative Evaluation 5.4.2 Quantitative Evaluation Limitations Summary Diffusion: Training a 3D Diffusion Model using 2D Images Background and Contributions	 66 66 67 70 72 73 75 77 81 81 82 83 83

8

	6.3	HoloDiffusion method						
		6.3.1 Learning 3D Categories by Watching Videos						
		6.3.2 Bootstrapped Latent Diffusion Model	88					
		6.3.3 Implementation Details	91					
	6.4	Evaluation	92					
	6.5	Limitations	94					
	6.6	Summary	95					
7	Holo	Fusion: Towards Photo-realistic 3D Generative Modeling	96					
	7.1	Background and Contributions	96					
	7.2	Introduction	97					
	7.3	HoloFusion method	99					
		7.3.1 HoloDiffusion revisited	100					
		7.3.2 HoloFusion	102					
	7.4 Evaluation							
		7.4.1 Details	106					
		7.4.2 Quantitative and qualitative analysis	108					
	7.5	5 Limitations						
	7.6	Summary	110					
8	GOI	Embed: Representation Agnostic 3D Feature Learning	111					
	8.1	Background and Contributions	111					
	8.2	Introduction	112					
	8.3	B Method						
		8.3.1 GOEmbed: Gradient Origin Embeddings	115					
		8.3.2 Experimental Evaluation Rubric	118					
	8.4	4 Plenoptic Encoding						
	8.5	3D Generation	121					
	8.6	Sparse-View 3D Reconstruction	125					
	8.7	Limitations	126					
	8.8	Summary	127					

9

		Contents	10
9	Con	nclusions	128
	9.1	Summary	128
	9.2	Insights	130
	9.3	What next?	133
		9.3.1 3D Meshes are important	133
		9.3.2 Scale is directly proportional to impact	134
Bi	bliog	raphy	136

Abbreviations	

172

Chapter 1

Introduction

3D is not special because it's "immersive", it's that 2D is deficient because it is not.

 \sim Amit Jain, Luma AI

1.1 Why 3D?

Computer Graphics is the systematic field of study of processes for creating 2D images with the use of a Digital Computer. The earliest Computer Graphics systems consisted of sophisticated routines for 2D drawings which aided various architectural and mapping applications [2]. These nascent graphics systems inspired almost all the operating system interfaces that we use currently. However those 2D graphics systems were not enough for more creative applications which required realistic shading and animations. Because of these niche requirements, very soon the discipline of Computer Graphics evolved to encompass 3D scenes, structures and motions. It was soon realised that it is much easier to simulate 3D rather than to emulate 3D in 2D [3]. Thus various graphics operations such as 3D scene structuring, coordinate systems, camera representations, camera and object transformations, and most-importantly 3D lighting and shading started receiving a rigorous mathematical and physics-based treatment [4, 5, 6]. This scientific treatment of the field of computer graphics not only pushed the development of various efficient and photo-realistic 3D algorithms, but also fueled creative industries such as Video Games and Movies (with CGI). Till today, Video Games and Movies remain the most widespread and the most important 3D creative applications.

It is commonly asserted that Gen-X grew up on reading books, Millennials on



Figure 1.1: (a) A screenshot from the highest grossing game of 2023 titled "God of War: Ragnarok", and (b) A frame from the 2022 blockbuster movie titled "Avatar: The way of water".

watching movies and *Gen-Z* on playing video-games. We can clearly see that with the progression of generations, the forms of entertainment as well as the forms of learning are inclining towards more and more "immersion". Today, the global cinema industry is estimated to be \$77 Billion [7] while the global video-game industry is estimated to be \$242.39 Billion [8]. Video-games inherently require 3D worlds and real-time interactions (fig. 1.1 (a)), but even movies today don't just reflect (or satire) the state of the current society; they push the horizons of human imagination through visual means powered by CGI [9, 10] (fig. 1.1 (b)). This underlines how socio-economically important and generally widespread these creative applications are; not to mention how important is 3D for driving these applications.

Apart from the traditional creative applications of Movies and Video-games, 3D is now being incorporated in the very nature of HCI (Human Computer Interface) itself. With the development of the VR (Virtual Reality) and AR (Augmented Reality) hardware, many more 3D applications such as 3D telepresence, and 3D infotainment are also on the rise. For all these applications, the interaction with the spatial-computing hardware has to be a lot more immersive and intuitive than our current screen-based computers. Since humans are used to the 3D world naturally, it is essential for the headsets to internally model the interfaces for a 3D world and not 2D screens.

Note that this discussion of 3D applications is far from exhaustive, and that the focus was mostly around creative 3D applications because that is where this thesis is directed towards. But, the requirement of 3D can be seen in many more applications, such as in fabrication (3D printing), medical imaging, manufacturing, authoritative previewing, etc. to name a few. In a nutshell, 3D is not only an additional good-to-have, but an essential feature of the future applications.

1.2 What is Generative Modelling?

As alluded to in the abstract, my thesis is about 3D Generative Modelling. Thus, we now take a slight detour here to discuss about GM (Generative Modelling) prior to highlighting the main goal and the scope of this thesis. Generative Modelling is a mathematical (and statistical) tool which imparts creative intelligence to computer algorithms. Given N i.i.d. samples $\{x^i\}_{i=1}^N$ from an *unknown* data distribution p(x), the task of generative modelling is to find the parameters θ of a parametric model $p_{\theta}(x)$ that best approximates p(x). Assuming an ideal model $p_{\theta}(x)$ that completely overlaps the density function of the true p(x), we could use it to draw synthetic samples $x' \sim p_{\theta}$, as well as compute the likelihood of newly observed (or doctored) samples. Apart from these two direct uses, it turns out that such a model acts as a foundational model for various other tasks. For instance, new posterior conditional models of the form $p_{\theta}(x|y)$ can be constructed using the foundational $p_{\theta}(x)$ such that y can be various different forms of correlated random variables that can act as conditioning. Also, since this hypothetical $p_{\theta}(x)$ model completely overlaps the true density function, we can assume that it implicitly understands the complete process of creation of x, and thus analysing it, perhaps in the form of directed cliques, potentially yields key insights regarding the creation process of x, which is usually intractable. These are only a few of the in-numerous points which underscore the importance of GM in Artificial Intelligence.

Although this mathematical definition of Generative Modelling sounds straightforward, it actually hides a lot of its complexity very deceptively. In almost all real-life applications the distribution p(x) over the random variable x involves modelling interactions between many underlying latent random variables $x_1, x_2, x_3, ...,$ leading to a non-factorizeable and highly intricate graphical model. Consider a simple example of the distribution of 3D chairs. Sampling a chair from the universal distribution over chairs involves understanding and simulating the process of making a chair; implicitly requiring to model infinitely many factors. For instance, the simple ones like the material, the colour, type, purpose etc. are factors that we can list, but there are many more complicated factors such as the manufacturing power, the socio-political scenario, the customs and traditions, etc. of the place where the chair is going to be created. As is the case with this simple example of making-chairs, it turns out that almost all the real-world processes have infinitely many underlying random variables, which makes them intractable to represent with probabilistic graphs.

The most popular solution to make this problem of Generative Modelling tractable is via data-driven ML (Machine Learning). The distribution p(x) is either explicitly or implicitly modelled by an ML model (typically a neural network) as a latent variable model that transforms a tractable and known probability distribution into the one being modelled. We cover the framework of ML, in detail, in the next chapter (chapter 2). As alluded to earlier, dissecting and uncovering the learned semantics of these models is then done as a post-training-analysis step.

1.3 What can Generative Modelling do?

One of the key characteristics of Generative Models is that there is no need for the creation of additional ground-truth manual labels for the data-points already in the dataset. As we will see in the next chapter, these types of models fall under the class of self-supervised machine learning methods. Further to this, Generative Modelling is an enabler of creative intelligence because it inherently tries to go beyond the tight and restrictive teachings provided by supervised learning regimes. For instance, an ML classifier might be able to classify two images as being a cat and an astronaut easily after training, but it won't be able to hallucinate¹ an image of an "astronaut-cat in space". Although imagining such a scenario is trivially intuitive for us humans, using only discriminative Machine Learning classification methods, such an imaginative model would be very difficult to train even after creating a

¹This hallucination on a classifier model could be performed using the gradient-based visualization algorithms such as DeepDream [11].



Figure 1.2: Samples generated by Stable-Diffusion [12] for the prompts (a) "A large public music concert on Mars"; (b) "A Unicorn swimming in the ocean"; (c) "Nean-derthals having a candle light dinner".

huge dataset of physically improbable images. Research to find other novel ways in which these generative models could be employed is ongoing, but, in this section, we highlight some of the most interesting applications that Generative Modelling has enabled in the process of 2D content creation, and the possible applications a 3D Generative Model could spawn.

1.3.1 Success stories of 2D Generative Modelling

Due to the readily available text and image based data on the internet, Generative Modelling has been studied extensively in these contexts leading to works such as LLMs (Large Language Models), DALLE [13], IMAGEN [14], Stable-Diffusion [12], and Giga-GAN [15]. These are breakthrough milestones that have successfully trained text-based auto-regressive and text-to-image based generative models on the scale of billions of data-samples respectively. Thus, having trained on these humon-gous sized datasets, the trained models can demonstrate unparalleled generalisation and outstanding ability for hallucination. Consider, for instance, the image-samples generated for highly imaginative text prompts by the Stable-Diffusion model in figure 1.2. The trained model has not just rotely memorized all the training-samples in the dataset, but has built an understanding of 2D textural concepts such as "the planet Mars" or "Unicorn" or "Neanderthals" or "candles" etc.; which it composes together to coherently generate the images on the canvas for the complete complicated text prompts.



Figure 1.3: Examples of live imaginative painting from the demo created by the X-user '@MartinNebelong'. The user is drawing an artistic tree branch in (a), while drawing a realistic portrait in (b). The ControlNet transforms the user's vector input into the specified styles of RGB images in real-time. Ref: https://x.com/AnimeshKarnewar/status/1759997147133444194.

Extending this further, the most-recent breakthrough from OpenAI [16] titled 'SORA' is the latest and the greatest text-to-video generation model. The model is capable of generating a minute of high fidelity video given text-prompts with unparalleled realism and text coherence. Although, scientific technical details have not been provided regarding the approach, a high-level official report reveals that the method is a variant of space-time latent-diffusion model trained on huge datasets using very large-scale GPU compute. Interestingly enough, the generated videos show remarkable 3D view-consistency even though an explicit 3D inductive bias was never applied on the model [17]. This points to the amazing potential that large-scale, monolithic, data-driven generative models hold.

Another line of research is currently figuring out ways in which these Diffusion Models can be fine-tuned to enable various creative controls. InstructPix2Pix [18] finetunes Stable-Diffusion to allow for instruction based text-editing. This has been incorporated in the latest ChaptGPT rendition which allows users to interactively create their art. More interesting is the work of Zhang et al. [19] titled ControlNet which fine-tunes the Stable-Diffusion model to allow for various spatial-conditioning apart from the text. Control-Net has spawned a myriad of applications including, but not limited to sketch-based realistic image generation, pose-to-image, depth-toimage, etc. Figure 1.3 shows an example of the real-time demos created by technical artists to boost their productivity using ControlNet.



Figure 1.4: Examples of textures generated by the Text2Tex method from Chen et al. [20] given pre-made 3D meshes and user provided text-prompts.

Lastly, it is remarkable to see that the text-to-image diffusion models are also enabling applications to increase the productivity of typical 3D content creation pipelines. Text2Tex, from Chen et al, [20] proposes an automated pipeline which can generate controllable textures for pre-made 3D meshes using Stable-Diffusion. The generated textures can be controlled using provided-text-prompts. Although this pipeline requires user created meshes, applications like Text2Tex clearly demonstrate how foundational the Stable-Diffusion generative model is. These applications could further benefit from leveraging the recent video-diffusion models in the short term, while core 3D generative models will contribute more significantly towards automating the process of 3D asset creation.

1.3.2 Possible applications of 3D Generative Models

The key ingredient that drives all the 3D applications is high-quality 3D content. Apparently, 3D content creation is much harder than 2D content creation, which either requires expensive and complicated scanning processes, or hours and hours of artistic modelling efforts. Considering the way these generative models have revolutionized the process of 2D content creation, we can fairly hypothesize that 3D generative modelling will also contribute significantly towards simplifying the process of 3D content creation. The disruption caused in the 3D industry by the automation of various aspects of the 3D content creation pipeline will not only make rolling-out new Video Games and Movies faster, but also ignite creativity among the artists since they will be able to focus more on the holistic aspects of the art rather than the low-level tools required for detailing. The Holy-Grail of 3D content creation is a system similar to the one used by the "*Memory-Maker*" in Blade Runner 2049 (ref: https://youtu.be/LKE4Bo7wRcY?si=9U4BgSq7gwNvQqMU). As shown in the sequence, the Memory-Maker uses a device reminiscent of the focal rings of a camera to create realistic 3D scenes. The minute details of the 3D world such as, intricate branches and leaves of the trees, style of the birthday-cake, dresses and hairstyles of the children, etc. are handled by the system; while the high-level abstract concepts are controlled by the Memory-Maker. Interestingly, by expanding the device rings, the system can also open-up in-depth control over something as trivial as the size of the head of the fly on a leaf, as shown in the beginning of the clip. Although, this system is part of science-fiction lore at the moment, it provides a vision for what automated 3D content creation pipeline needs to realise in the future. And, similar to the live imaginative 2D painting demo of figure 1.3, a large-scale data-driven 3D generative model is exactly what will enable such a 3D content creation system.

The current offline 3D rendering engines simulate the process of light-transport to synthesize 2D images almost indistinguishable from reality. However, they require a very long time in order to do so, restricting their use in offline applications such as Movies, Advertisements, etc. Apart from the time constraint, there are various phenomena such as wave-optical effects, diffraction, pearlescence etc. that even the most sophisticated offline 3D renderers cannot model correctly because we do not know the physics behind them yet. The online renderers can produce 2D images much faster to enable real-time interactive applications such as 3D Video Games, but, they inhibit many photorealistic effects and are limited to only hyperreal or artistic 3D scenes. A 3D Generative Model that has been trained on billions of 3D assets will implicitly model all the common and rare physical effects, various real, hyperreal, and synthetic styles, and be able to produce 3D samples in real-time with efficient implementation. Interestingly, such a model could serve as a bridge between the most sophisticated online and offline rendering systems. Provided with a description of the 3D scene, a 3D Generative Renderer will be able to synthesize photorealistic renders in real-time.

Similar to what we saw in the previous subsection, the 2D text-to-video model already contains 3D information, and is applicable in the 3D application of texturing assets; the 3D generative model will also be able to go beyond just 3D. A large-enough 3D generative model trained on static 3D assets could be leveraged to generate realistic animations as well. Although slightly philosophical in nature, but the 3D generative model might also uncover certain insights in our possible 4D world; for instance uncovering a hyperspace guiding the creation process of a particular class of 3D objects.

1.4 Scope of the thesis

With the discussion in the previous sections, where I motivated why 3D is important and what Generative Modelling can do, I now describe what this research thesis endeavours.

Although the holistic problem of Generative Modelling is far from being solved, especially with new generative modelling methods being proposed even now, for example Denoising Diffusion GANs [21], VARs [22], and FlowMatching [23] models; a thesis to contribute to Generative Modelling as a whole is too broad to be taken seriously in today's research scenario. Hence I narrow down the scope of this thesis into a realisable goal such that small tangible steps can make substantial contributions towards its realisation.

Firstly, I will only be concerned with applying generative modelling to 3D data. This includes both synthetic data such as artistically modeled 3D meshes, their textures, physically based materials, etc., as well as the real-captured 360° image-based and video-based data of real-world objects. Although animations depicting realistic motions are a key characteristic of 3D applications, I will only be working with static 3D assets in this body of work. I believe that considering motions, and other physical effects of 3D objects forms an orthogonal line of research and that the exclusion of dynamic 3D assets helps defining a more realisable goal.

Secondly, 3D generative models that are practically implementable in code and

executable on the present GPU hardware are what will make the most scientific and social impact. Computational efficiency, both in terms of training and inference, plays a key role in the adoption of proposed generative modelling methods. For instance, Probabilistic-Flow based models stagnated due to their extremely long training times and Diffusion Models were very slow in adoption due to their slow inference. Hence, instead of only theorizing algorithmic models, I will also take into consideration the Computational Efficiency of the 3D generative models as a key guiding principle.

Thirdly, we know that visual quality of the generated samples is very important in case of generative modelling. For instance, GANs were preferred over VAEs and AutoRegressive models because they could generate compelling image samples even though they are extremely hard to train. Thus, the quality of the samples produced by the Generative Model is one of the key measures for assessing its potential. In case of 3D scenes and assets, photorealism is the ultimate benchmark of visual quality, and thus I aim towards generative model that can produce photorealistic 3D samples.

And finally, the breakthrough 3D generative model will have to be trained on billions and billions of real-captured as well as synthetic static 3D assets. Thus scalability of the 3D generative modelling methods is the last piece in the puzzle of 3D generative modelling.

In summary, the scope of this thesis is to contribute substantial strides towards a specific class of Generative Models, i.e.,

Towards Computationally Efficient, Photorealistic, and Scalable Generative Modelling of static 3D assets.

1.5 Challenges and Opportunities

There are two main challenges towards achieving 3D generative models. Firstly, there is a lack of high-quality large-scale 3D dataset; secondly, unlike 2D images, which are represented as grids-of-pixels, 3D assets do not have a de-facto digital representation; these challenges are discussed in detail in the following paragraphs.

Towards the first challenge, recently, various data collection efforts such as



Figure 1.5: Figure shows 2D orthographic projection of three 3D representations namely point-cloud (left), mesh (middle) and voxel-grid (right).

Objaverse and Objaverse-XL [24, 25] are gathering millions of 3D textured-mesh based assets. Also similar to Shutterstock, it can be expected that more and more Gaming-studios and VFX-studios will democratize their 3D assets for progressing the research in 3D generative modelling. More interestingly, the process of 3D asset creation is already being aided with the use of 2D text-to-image generative models. As discussed earlier, works like Text2Tex [20] can speed up the process of 3D texture/material painting over pre-made meshes. Whereas, methods such as DreamFusion [26] which can distill 3D assets from the 2D text-to-image generative models such as Instant3D, SyncDreamer, etc. [28, 29, 30, 31] promise a novel bootstrapped approach to forming a large-scale dataset of 3D assets. In a nutshell, more and more synthetic 3D can now be created much faster using 2D pretrained text-to-image models than earlier; which will aid overcoming the first challenge of lack of 3D data. However, collecting 3D data of real scenes still remains an open challenge.

Secondly, since there is no de-facto representation for 3D data that is universally used for all applications, different 3D representations like point-clouds, meshes, voxel-grids, etc. are used depending on the application in which they are deployed. As shown in figure 1.5, point-clouds simply denote the various points either on the surface of the 3D object or inside a volume for certain applications. Point-clouds are mostly used for sensing/understanding-building applications (like robotics, autonomous driving, etc.) due to their ease-of-use and low computation/processing requirements. However, point-clouds give very little visual information and are rarely used in visual applications such as CGI or video-games. Meshes provide more information about the surface geometry of the 3D objects by explicitly denoting the connectivity information on top of the point-clouds. Majority of the 3D graphics/rendering systems make use of triangle meshes with support for hardware acceleration for their processing. With high-enough resolution, and using state-of-the-art rendering pipelines, meshes can represent the 3D scenes at almost photo-realistically indistinguishable quality. A much simpler representation, and more parallel to the 2D raster image grids, is voxel-grids. Voxel grids store the 3D data such as the density information (or occupancy) and/or various scattering parameters and/or view-dependent out-going radiance on a regular 3D grid. They are much easier to handle in code but are very restrictive in terms of their memory consumption for storage as well as processing. We cover in the next chapter the volumetric representation called 3D Radiance fields which we use in our subsequent projects (chapters) because of it's amenability to our needs. However, apart from these data-point-based discrete representations, there has been a surge in the research related to neural 3D scene representations, called Neural Fields. The Neural Fields movement in the 3D data representations research was spear-headed by the works such as OccupancyNetworks [32] and SRNs [33] that proposed to use MLPs for representing the 3D shapes as occupancy fields and SDFs (signed-distance fields) respectively. But, most notably, the work NeRF (Neural Radiance Fields) [34] made this idea popular by demonstrating the use of MLPs in the context of an application as complex as 3D Novel-view-synthesis. Concurrently, another important work called SIREN [35] showed that sinusoid-activated MLPs could also represent images, videos, MRIs, etc. The research progress exploded after these two works, leading to various works and tremendous research interest in neural 3D scene representations. I cover more of the related work in the chapter 3.

Chapter 2

Preliminaries

The research works presented in this thesis, viz. chapters 4, 5, 6, and 7, require some basic understanding of: **Deep Learning**, **3D Radiance Fields**, and **Generative Models** such as **GANs** and **Diffusion** models. I cover a brief overview of these concepts in the following sections, and point the reader to relevant resources for more in-depth explanations.

2.1 Deep Learning

There are various problems in CS (Computer Science) that have been studied continuously over the last six decades. We can perceptually divide the problems into humanly-intuitive and non-intuitive problems¹. The problems from the non-humanly intuitive categories such as sorting, shortest-path-finding, knapsack, etc. have algorithmic solutions which can yield correct answer for each and every instance of these problems. Counter-intuitively, the humanly-intuitive problems such as recognizing images of objects, translating from one language to another, etc. are extremely hard for computers to solve [37]. And thus to make progress on the class of humanly-intuitive problems, heuristic approaches were studied extensively. Heuristic approaches differ from algorithmic approaches in that they only provide a correct solution in a specific percentage of instances, as opposed to solving all the instances. The heuristic approaches provide a probabilistic outlook on solving these problems and are preferred for two reasons. Firstly, the heuristic approaches were the

¹Note that this is not a formal categorization like the division from Theory of Computation [36].



Figure 2.1: A Venn diagram showing where **Deep Learning** fits in the hierarchy of the concepts in AI.

only ones that sort-of worked for these highly difficult humanly-intuitive problems and allowed the resarch(ers) to make progress. And secondly, they are preferred over algorithmic approaches for these problems because of the inherent ambiguity present in these problems; for instance, two images of different objects may look extremely similar to each other causing both class-predictions to be acceptable, or an image may be too corrupted by low-lighting such that a classification may not be possible at all.

The technique majorly employed for solving such problems turns out to be **ML** (**Machine Learning**). Figure 2.1 shows the hierarchy of fields and sub-fields in AI, highlighting where **Deep Learning** sits inside the broader class of Machine Learning; which consists of methods which enables computers to solve heuristic humanly-intuitive problems by learning from a number of data examples. A typical machine-learning framework consists of four components (see fig 2.2):

- 1. A parametric model to capture the learned experience.
- 2. An **objective function** which specifies what the parametric model is supposed to learn.
- 3. A training routine which describes the optimization and learning loop.
- 4. Last but the most important, the data used for training.

Machine Learning framework



Figure 2.2: The figure describes the flow of information in a typical Machine Learning framework. Both the example_input and example_output are the same for self-supervised learning, while in the case of unsupervised learning (K-means for instance), the objective function doesn't take as input any example_output.

Deep Learning is a sub-class of Machine Learning which uses the overall framework of Machine Learning, but specifically introduces a class of the **parametric models** which are inspired by the biological structure of brain-cells (neurons) called **Neural Networks**. As a result of using neural-networks as parametric models for ML, the model internally learns a **deep** hierarchy of concepts in which the more complex concepts are built on top of the simple ones, hence the name, **Deep Learning** [37]. We direct the interested readers to Goodfellow et al [37] for more in-depth coverage of Deep Learning. In the next few subsections, we now describe the details of the relevant components used in the **Deep Learning** framework.

2.1.1 Neural Networks

Majority of the success in ML has come from using neural-networks as the models (i.e. Deep Learning) and using supervised learning algorithms. The classification of ML methods into being supervised, unsupervised and self-supervised is not very specific, but generally, methods which use labelled input-output data pairs are called as supervised learning methods, while the ones which don't are either unsupervised or self-supervised. The distinction between unsupervised and self-supervised has to do with how the algorithm actually learns. In case of self-supervised algorithms the example_input data is itself used as labels to train models such as AutoEncoders



Figure 2.3: The perceptron algorithm (left) was the basis of the modern-day neural-network models used today. (Right) shows how the individual perceptrons are arranged in case of a Multi-Layered Perceptrons model.

or other types of generative models, while the unsupervised algorithms have an objective function which doesn't involve the use of labels at all, for instance clustering algorithms such as K-Means, K-Nearest Neighbours, LDA etc. The Neural Networks are mostly (almost exclusively) used in the context of supervised or self-supervised learning methods. Here we describe the most influential and now most commonly employed forms of Neural Networks in Deep Learning. Our discussion of these ML model architectures is agnostic of the method of optimization used to train them. The mathematical optimization techniques are covered in the next subsection.

The basis of the Neural-network models, the **Perceptron** algorithm (refer fig 2.3), dates back to 1957 in which Rosenblatt [38] introduced a very simple but automatically learnable binary classification model. The Perceptron algorithm takes as input a vector $x = [x_0, x_1, ..., x_n]^T$ and outputs a binary classification $\hat{y} \in \{0, 1\}$. The function that the perceptron learns can be described as:

$$\hat{y} = P_{\text{tron}}(x; w, b)$$

= $\sigma(w.x+b)$

Where $w = [w_0, w_1, ..., w_n]^T$ and *b* form the learnable parameters of the algorithm called as the weights and the bias respectively. The $\sigma : \mathbb{R} \to \mathbb{R}$ is the activation

function (in this case, the step function). A single perceptron can only learn linearly separable patterns, while is unable to approximate non-linear classifiers. However, later works in the following years [39, 40, 41] realized that stacking up multiple perceptrons to form MLP s (see fig 2.3) can learn any continuous non-linear function. This line of research piqued when Rumalhart et al proposed an end-to-end pipeline for training the MLP s using **Backpropagation** and the **Stochastic Gradient Descent** optimization. Various non-linear activation functions can be employed in the MLP custom fitting the applications and/or data-domain, but today, the Rectified Linear Unit (ReLU) function is synonymous as an activation function with MLPs due to it's various appealing characteristics.

It can be observed that for various problems in high-dimensional data domains such as image-recognition, image-segmentation, etc., or for temporal domain tasks such as text-recognition, text-classification, text-completion, etc. the MLP architecture is not feasible due to the explosion of number of parameters. In order to be able to solve these problems, special architectures have been devised to introduce the inductive biases based on the domains of the input. For images, it was realised that a fully-connected MLP is an overkill, since the visual concepts depicted in images have translational invariance, and scale and rotational equivariance for discriminative tasks such as classification and semantic segmentation. In simpler words, an image will be classified in the cat class irrespective of where in the image the cat appears, how big the cat is, and if the cat is upside-down or not. A new class of NNs named CNNs (Convolutional Neural Networks) [42] were introduced for image based problems. CNNs are today the de-facto architecture used for any image domain problem, right from classification, to semantic segmentation, to even 2D Image generation. The progression of CNNs went from the first proposed LeNet architecture [42], by LeCun et al, till the ResNet architecture [43]; which became the state-of-the-art in Image recognition. Some of the notable works in this progression include AlexNet [44], VGGNet [45], InceptionNet [46], and SE-Net [47] which introduced various novel architectural modalities in the CNN operations, but keeping the core convolution operation the same. We note that there were actually many more works along the way, which are out of scope to be covered here. The main idea of the CNN is to replace the vector-dot product operation between the inputs x and the weights w of a single perceptron (more commonly referred to as a neuron now) by the convolution operation. Thus, a single neuron of the CNN computes the following function:

$$\hat{y} = CNN(x; w, b)$$

= $\sigma(x * w + b)$

Where, both *x* and \hat{y} are now 2D instead of 1D vectors, and the weight *w* is much smaller in dimension compared to the inputs. This operation can be viewed as learning the kernel of a convolutional image filter. However, the neural-convolution operation doesn't provide any dimensionality reduction which is required for sparse prediction tasks such as classification. Thus to introduce dimensionality reduction and to provide scale-invariance to some extent, the pooling operations are used by CNNs. A typical pooling operation *Pool* : $\mathbb{R}^{c \times h \times w} \to \mathbb{R}^{c \times h/2 \times w/2}$ allows only certain activations to pass through it, killing all the non-required spatial information. Please refer to the chapter 9. of Goodfellow et al [37] for an in-depth discussion of Convolutional Neural Networks.

Temporal domain NNs are out of the scope for our thesis, but for completeness of the discussion, we briefly cover the two most influential variants, viz. RNN s [48] and Transformers [49] here. RNN s or Recurrent Neural Networks operate in the temporal domain and can be described by the following equation:

$$x_t, h_t = RNN(MLP(x_{t-1}), MLP(h_{t-1}); w_{rnn}, b_{rnn})$$

Where, x_t , h_t , x_{t-1} , and h_{t-1} are arbitrarily shaped vectors. The *h* vectors record and specify the hidden state for an input *x* at any particular instant in time. The RNNs are trained by unrolling a truncated number of steps in practice.

The Transformer architecture became the state-of-the-art algorithm for various language, speech and other temporal domain problems. At the core, it introduced a

novel operation called self-attention which can be described as follows:

$$\hat{y} = SA(Q, K, V)$$

$$= \operatorname{softmax}\left(\frac{QK^{T}}{\sqrt{d_{k}}}\right)V$$
where, $Q = f_{q}(x); K = f_{k}(x); V = f_{v}(x)$

Note that here the input *x* and the output *y* are vector sequences thus, $x, y \in \mathbb{R}^{d_k \times T}$, where d_k is the vector dimensionality and *T* are the number of time-steps. The Self-Attention operation basically computes an affinity between each query and key vector to then finally obtain the value-vectors through the softmax operation, which highly resembles the attention operation of the Neural Turing Machine [50]. Interestingly, all the three components, Q, K, and V come through functional mapping from the input, hence the name self-attention. Transformers have apparently turned out to be much more universally applicable and today form the basis of not only temporal domain, but various other spatial and non-spatial domain problems as well. We recommend the original paper on Transformers by Vaswani et al. [49] and the tutorial videos by Grant Sanderson [51, 52] for an in-depth explanation of Transformers.

2.1.2 Mathematical Optimization

Training an NN can be viewed as an optimization process that minimizes or maximizes the criterion provided by the objective function to tune all the learnable weights and biases of the NN. The objective function typically measures how well the NN s output, for a given particular input, matches the ground-truth output from the dataset. Covering all the different possible optimization strategies such as genetic methods, swarm optimization, gradient based methods etc. is out-of-scope, and thus, we only focus on the specific gradient based method called stochastic gradient descent here. There is another ingredient known as the **Backpropagation algorithm** which computes the gradients of the objective function wrt. the NN parameters which is required for the SGD to work. We discuss both Backpropagation [48] and SGD [53] as follows:

2.1. Deep Learning

The Backpropagation algorithm [48] has been one of the most seminal contributions that allowed NNs to be trained autonomously on computers. At the core, the Backpropagation algorithm makes use of the simple concept of *chain-rule* of differentiation for computing the gradients. For a general objective function $\mathcal{L}(\hat{y}, y)$, where $\hat{y} = NN(x)$, the backpropagation algorithm enables the computation of the gradients $\nabla_{\{\Theta,\beta\}}\mathcal{L}$, where Θ and β represent the set of all the trainable weights and the set of all the trainable biases of the neural network *NN* respectively. A typical neural network will compute the output \hat{y} as a series of compositions as follows:

$$\hat{y} = NN(x) = \sigma(nn_n(\dots\sigma(nn_2(\sigma(nn_1(\sigma(nn_0(x;\theta_0,b_0));\theta_1,b_1));\theta_2,b_2)));\theta_n,b_n)$$

We can write this composition as a series of equations by introducing new variables for the unactivated and activated hidden-vectors as follows:

$$h_0 = nn_0(x; \theta_0, b_0)$$

$$a_0 = \sigma(h_0)$$

$$h_1 = nn_1(h_0; \theta_1, b_1)$$

$$a_1 = \sigma(h_1)$$

$$\vdots$$

$$\hat{y} = \sigma(h_n; \theta_n, b_n)$$

The Backpropagation algorithm then proposes to apply the chain rule to compute the gradients for any parameter θ_k and b_k as follows:

$$\frac{d\mathcal{L}(\hat{y}, y)}{d\theta_k} = \frac{d\mathcal{L}(\hat{y}, y)}{d\hat{y}} \cdot \frac{d\hat{y}}{dh_n} \cdot \frac{dh_n}{da_{n-1}} \cdot \frac{da_{n-1}}{dh_{n-1}} \cdots \frac{dh_k}{d\theta_k}$$
$$\frac{d\mathcal{L}(\hat{y}, y)}{db_k} = \frac{d\mathcal{L}(\hat{y}, y)}{d\hat{y}} \cdot \frac{d\hat{y}}{dh_n} \cdot \frac{dh_n}{da_{n-1}} \cdot \frac{da_{n-1}}{dh_{n-1}} \cdots \frac{dh_k}{db_k}$$

Using this algorithm, the gradients $\nabla_{\{\Theta,b\}}\mathcal{L}$ can be calculated for weights and biases

of the NN in one single pass. This gradient computation pass can be viewed as propagating the error/loss value through the NN in the backward direction, and due to this view, the name of the algorithm has been proposed to be **Backpropagation**.

Algorithm I Stochastic Gradient Descent							
Require: <i>m</i> number of training iterations, <i>D</i> dataset, <i>NN</i> neural network for training							
with Θ and β as the trainable weights and biases, and α learning rate for SGD							
1: for $i = 0, 1, 2,m$ do							
2: $(X_{\text{batch}}, Y_{\text{batch}}) \sim D$ \triangleright sample random batch from D							
3: $\hat{Y}_{\text{batch}} := NN(X_{\text{batch}}; \Theta, \beta)$							
4: $\hat{\Theta} := \nabla_{\Theta} \mathcal{L}(\hat{Y}_{\text{batch}}, Y_{\text{batch}}) \triangleright \text{ compute and backpropagate errors for weights}$							
5: $\hat{\beta} := \nabla_{\beta} \mathcal{L}(\hat{Y}_{\text{batch}}, Y_{\text{batch}}) $ \triangleright compute and backpropagate error for biases							
6: $\Theta := \Theta - \alpha \hat{\Theta}$ > SGD update equation							
7: $\beta := \beta - \alpha \hat{\beta}$ > SGD update equation							
8: end for							

The **Stochastic Gradient Descent** (**SGD**) algorithm is a gradient based optimization method used extensively to train NN and ML models. Given a dataset $D = \{(x^0, y^0), (x^1, y^1), ...(x^i, y^i)\}$ (where |D| = N) of *N* input-output pairs, SGD runs *m* iterations of the optimization loop in which first a batch of input-output pairs is randomly sampled, followed by running the forward pass of the network and computing the loss of the predicted output wrt. the output sampled from the dataset, followed by running the backward pass to backpropagate the errors and then finally using the SGD update equation (line 6 and 7 of alg. 1). This enables the NN to tune the parameters such that the loss is minimized, thus allowing the NN to progressively better map the input to the output given the dataset. Algorithm 1 describes this procedure in the form of a pseudo-code.

2.2 3D Radiance Fields

The most widely deployed production pipelines for 3D rendering software use

- 1. 3D meshes to represent various geometries (w/ or w/o motion),
- 2. texture-maps or BRDF-maps [54] to represent the materials,
- 3. and meshes or environment-maps for representing light-sources [4].

Due to which, most of the 3D visual-application based data found today is either in the form of 3D meshes or texture/brdf/environment maps, utilised in the context of either non-physically based (real-time) or physically based (offline) renderers. Thus, the logical next step is to research ways in which can we learn, from observations, generative modelling distributions over these. However, as pointed out in the comprehensive surveys by Tewari et al [55, 56], the mesh-based 3D representations are hard to utilise in the context of large-scale generative learning due to the inherent discreteness, sparsity and high-sensitivity to initialization. Another challenge with directly using the mesh based representations is that high-quality realistic rendering of these is very expensive with regards to compute and resources. And, in order to avail a very large scale of these mesh-based 3D assets (of the order of Billions similar to the 2D generative models), methods for automatic reconstruction of meshes from multi-view imagery are being heavily researched. Although significant progress has been made towards reconstructing 3D meshes or discrete 3D representations [57, 58] via differentiable mesh rendering (soft-rasterization [59, 60]), obtaining high-quality 3D geometries *directly* through inverse rendering is a formidable challenge. This restricts the creation of 3D meshes to manual crafting by artists or through expensive scanning procedures which also need manual fine-tuning most of the time.

Unlike the Surface-based 3D mesh representations, 3D Volumetric representations are much easier to optimize given image observations since the gradients available for them during optimization are dense and informative, while also being robust to random-initialization strategies. A **3D Radiance Field** is a volumetric representation which captures not only the 3D geometry (in the form of density), but also photo-realistic appearance (in the form of view-directional emitted radiance) of the 3D objects/scenes. Since the appearance is encoded in the form of emitted radiance, the volumetric 3D radiance fields cannot be re-lit through newly added dynamic light sources. Although methods have been proposed to extract the traditional mesh and BRDF based assets from optimized radiance fields [61, 62], research towards transforming 3D radiance fields into 3D scattering fields is heavily ongoing. Nevertheless, the Radiance Fields are a popular and quite useful representation due to their peculiar aforementioned benefits over 3D meshes. We discuss 3D Radiance Fields in detail as follows:

Mathematically, the 3D Radiance Field \mathcal{R} is a continuous function (A 4D vector field in a 5D space) $\mathbb{R}F : \mathbb{R}^5 \to \mathbb{R}^4$ which maps a 3D location $p = [x \ y \ z]^T$ and an exitant viewing direction $d = [\theta \phi]^T$ to a scalar volumetric density σ and the colour in the form of tri-wavelength radiance $c = [r g b]^T$. There are various ways of parametrizing this vector field, but some of the most influential representations include: Mildenhall et al [34] which uses a single NN; Lombardi et al [63], Karnewar et al [64], Sun et al [65] which use different flavours of 3D voxel grids; Muller et al [66] which use neural feature based hash-grids; and Chan et al [67] which use feature triplanes. We note that there are many more works which propose ways of parameterizing the 3D Radiance fields in novel hybrid ways, and direct the interested readers to the state-of-the-art survey on Neural Fields by Xie et al [68]. We note that the most important aspect of the 3D radiance field parameterization for us is it's amenability to generative modeling, and thus we pick-and-choose the one which fits our need for that particular project. Specifically, we use ReLU Fields [64] in 3inGAN [69] (chapter 5) and Voxel-feature grids in HoloDiffusion [70] (chapter 6) and HoloFusion [71] (chapter 7). Considering this disarray of these parameterizations, we propose a method for encoding 2D image views into arbitrary 3D representation of 3D radiance fields in GoEmbed (chapter 8).

2.2.1 Differentiable volumetric rendering

As alluded to in the earlier discussion, we know that the 3D radiance fields are static 3D assets which are primarily used for photo-realistic rendering. Thus here we discuss how the 3D Radiance field assets are rendered. More importantly, this rendering operation is continuous, smooth and differentiable, and hence can be used in the optimization setup allowing to learn the 3D radiance fields from 2D image observations. We discuss this end-to-end 3D reconstruction pipeline in the next subsection. Without loss of generality, let \mathcal{R} denote the 3D radiance field, Thus, for each 3D point $p = [x \ y \ z]^T$ in the domain of \mathcal{R} and for any arbitrary viewing direction $d = [\theta \ \phi]^T$ on the sphere SO(2), \mathcal{R} maps them to a 4D vector $f = [\sigma \ r \ g \ b]^T$.



Figure 2.4: Differentiable volumetric rendering of a continuous volumetric 3D radiance field denoted by \mathcal{R} .

The rendering operation $\zeta(\mathcal{R}, P)$ yields a 2D image from the camera pose *P* for the radiance field \mathcal{R} . The camera pose *P* is represented by a 4 × 4 homogeneous matrix containing the 3 × 3 rotation matrix for the orientation of the camera, and the 3 × 1 translation vector for the location of the camera. Given the camera centre (the translation vector) *o* and the viewing direction vector² *d*, we define the rendering ray as r(t) = o + dt, where $t_{\text{near}} < t < t_{\text{far}}$ denotes the depth of the point from the camera centre. For the ease of the integration, we can denote the vector-field \mathcal{R} as a tuple: $\mathcal{R}(r(t), d) = (\sigma(r(t)), c(r(t), d))$. Note here that the density is not dependent on the viewing-direction; the density of a physical object doesn't change based on the direction from which we perceive it, whereas the appearance indeed does. The colour value for the particular ray *r*, and hence for the particular pixel, is computed

²the viewing direction vector is computed by subtracting the o vector from the pixel location on the camera's imaging plane.

using the Emission-Absorption [72, 73] ray-marching operation as follows:

$$c^{r} = \int_{t_{\text{near}}}^{t_{\text{far}}} T(t) \alpha(r(t)) c(r(t), d) dt$$

where, $T(t) = e^{\left(-\int_{t_{\text{near}}}^{t} \sigma(r(s)) ds\right)}$
and $\alpha(r(t)) \equiv 1 - e^{\left(\sigma(r(t)) dt\right)}$

Intuitively enough, repeating this integration for all rays in a camera P yields the rendered 2D image; given that the intrinsic properties of the camera, such as the height, width, and the focal_length are specified. Since continuous integration of arbitrary functions is not possible on computers, we instead use quadrature for approximating them. In spite of being an approximation, this discrete version is quite useful because it provides the integration for the whole continuum of the domain (i.e. all points on the ray between t_{near} and t_{far}). This is done as follows. We first sample values of t_i on the ray from N stratified bins using linear interpolation between the chosen bin limits t_{near} and t_{far} :

$$t_i \sim \mathcal{U}\left[t_{\text{near}} + \frac{i-1}{N}(t_{\text{far}} - t_{\text{near}}), t_{\text{near}} + \frac{i}{N}(t_{\text{far}} - t_{\text{near}})\right]; i \in [1, N]$$

Then we compute the approximation of the integral as follows:

$$\hat{c}^r = \sum_{i=1}^N T(t_i)(1 - e^{-\sigma(r(t_i))\delta_i})c(r(t_i))$$

where, $T(t_i) = e^{-\sum_{j=1}^{i-1}\sigma(r(t_j))\delta_j}$

The values $\delta_i = r(t_{i+1}) - r(t_i)$ denote the physical distance between any two adjacent samples *t*. The use of delta allows the approximation to have the full continuous domain. Thus, in practice, even if slightly perturbed *t* locations get sampled per-ray per rendering, the rendered image doesn't have much variance, if a sufficiently high number of samples are chosen. Most importantly, the rendering equation is trivially differentiable and can be back-propagated through using off-the-shelf auto-grad packages such as PyTorch.

2.2.2 End-to-end 3D reconstruction pipeline

Neural Radiance Fields, by Mildenhall et al [34], popularized the end-to-end 3D reconstruction pipeline using only 2D Image observations. Another most notable work during that period is Neural Volumes, by Lombardi et al [63], which also had a similar optimization pipeline. While Neural Volumes proposed to learn a Variational AutoEncoder (VAE) over a domain of 3D scenes (specifically 3D human faces), NeRF only restricted the optimization to a single 3D scene. We note that there have been countless works since then that have made novel contributions to this pipeline. The survey titled "NeRF-Explosion" covers this vast array of works in detail [74]. In this section we describe the vanilla version of this pipeline succinctly.

Input to this pipeline are a set of images $\mathcal{I} = \{I_1, \ldots, I_n\}$ and corresponding camera poses $\mathcal{C} = \{C_1, \ldots, C_n\}$, where each camera pose consists of $C = \{R, H, W, T, F\}$; R is the rotation-matrix ($R \in \mathbb{R}^{3\times3}$), T is the translation-vector ($T \in \mathbb{R}^3$), H, Ware the scalars representing the height and width respectively, and F denotes the focal length of the assumed simple pin-hole camera. We assume that the respective poses for the images are known either through hardware calibration or by using structure-from-motion [75]. We denote the rendering operation to convert the 3D radiance field \mathcal{R} and the camera pose C into an image as $\zeta(\mathcal{R}, C)$. Thus, given the input set of images \mathcal{I} and their corresponding camera poses C, the problem is to recover the underlying 3D radiance field \mathcal{R} such that when rendered from any $C_i \in C$, \mathcal{R} produces rendered image \hat{I}_i as close as possible to the input image I_i , and produces spatio-temporally 3D consistent \hat{I}_j for poses $C_j \notin C$.

• • •	A	١ţ	gori	itl	hm	2	3D	rad	liance	field	reco	nstr	uctic	n p	oipe	line
-------	---	----	------	-----	----	---	----	-----	--------	-------	------	------	-------	-----	------	------

Require: <i>m</i> number of training iterations, $D = \{(I_i, C_i) I_i \in \mathcal{I} \text{ and } C_i \in \mathcal{C}\}$ dataset.								
${\cal R}$ learnable 3D radiance field rep	presentation, and α learning rate for SGD							
optimization.								
1: for $i = 0, 1, 2,m$ do								
2: $(I_{\text{batch}}, C_{\text{batch}}) \sim D$	\triangleright sample random batch from D							
3: $\hat{I}_{\text{batch}} := \zeta(\mathcal{R}, C_{\text{batch}})$								
4: $\mathcal{L} := \hat{I}_{\text{batch}} - I_{\text{batch}} _2^2$	compute photometric loss							
5: $\hat{\mathcal{R}} := \nabla_{\mathcal{R}} \mathcal{L}$	▷ backpropagate loss to radiance field							
6: $\mathcal{R} := \mathcal{R} - \alpha \hat{\mathcal{R}}$	▷ update the radiance field via SGD							
7: end for								
Algorithm 2 describes the pseudo-code of this optimization pipeline. In simple words, the optimization pipeline uses the overall framework of ML training, but replaces the ML model by the radiance field \mathcal{R} , and uses the ground-truth image-observations with their corresponding poses as the data for training. The gradients are computed via backpropagation and optimization is done using SGD without any modifications. In chapter 4 we propose a novel 3D representation for the radiance field \mathcal{R} titled ReLU-Fields [64], which is a very thin neural extension of the traditional voxel grids that allows to capture sharp high-frequency details in the 3D geometries of the scenes being optimized. Please refer this chapter for examples of how well this pipeline works for 3D scene reconstruction.

2.2.3 Summary

3D Radiance fields are a formidable alternative to 3D textured/materialed meshes, which shines the most in the context of end-to-end 3D scene reconstruction pipeline. Since our primary goal is to learn the generative distributions over 3D assets, 3D Radiance fields suit our needs as follows:

- They can be optimized (reconstructed) from 2D multi-view images at scale. For example the Co3Dv2 dataset [76].
- 2. The trivially differentiable rendering of the radiance fields allow us to supervise our generative models directly from Images and Poses.
- Various representations for 3D radiance fields such as ReLU-Fields [64], Feature voxel grids [70] and Triplanes [67] can be input naturally to 3D generative model architectures.

Lastly, having restricted the scope of this thesis to static, non-relightable, objectcentric 3D assets, 3D Radiance Fields are perfect in our research context. We use them as the primary representation for the projects described in the subsequent chapters.

2.3 Generative Models

The proposed methods for generative models so far can be categorized into three classes:

- 1. Implicit density models: GANs
- 2. Likelihood based models: Autoregressive, VAEs, Normalizing flows, and the latest Diffusion Models.
- 3. Energy based models: RBMs, DBNs.

Since, GANs are required for 3inGAN (chap. 5) and Diffusion based generative models are required for HoloDiffusion (chap. 6), HoloFusion (chap. 7), and GoEmbed (chap. 8), we cover these two generative models here in brief. Please refer to the comparative survey, by Bond-Taylor et al [77], for an overview of the different types of generative models.

2.3.1 Generative Adversarial Networks

The GANs (Generative Adversarial Networks) model *belongs* to the implicit density models. GAN only learns to model the sampling procedure of a complex distribution and does not allow for computation of explicit probability density values of samples or the likelihood of observed samples. Introduced first in 2014 by Goodfellow et al. [78], GANs were the state-of-the art generative models for a long time till Diffusion Models took over. The sample quality that GANs could reach at the time was not attainable by other competitor methods such as VAEs, and Flow-based models. The idea behind the GAN model is explained as follows, while we cover Diffusion Models in the next sub-section.

Given a generator network G(z) which maps a random latent variable distributed according to a known tractable prior distribution (Gaussian is the de-facto choice) to the desired random variable x, the problem of implicit density generative modelling boils down to defining a distance function $d(p_{\theta}, p)$ between the modelled p_{θ} and the ground-truth p distributions. Interestingly, the concept of L2 distance breaks on very high-dimensional spaces such as the space over 100×100 pixel images. Consider



Figure 2.5: Two perceptually very similar images \mathcal{I}_1 and \mathcal{I}_2 have the maximum possible L2 distance in the 4×4 image space where each pixel can have values strictly between [0, 1], s.t. 0 represents the colour black and 1 represents the colour white.



Figure 2.6: The interaction between the two networks, viz. Generator G(z) and the Discriminator D(x) (for real sample input) or $D(\hat{x})$ (for fake sample input) during mini-max game of the GAN training.

the example images \mathcal{I}_1 and \mathcal{I}_2 as shown in the figure 2.5. The two images are perceptually indistinguishable (effect increases with resolution), but the *L*2 distance penalizes them to be far apart. Thus, the conventional distance functions such as the *L*2 distance over euclidean spaces cannot be used in this context, and an ingenious solution is required.

Some of the first approaches tried to model this with moment matching [79],

but suffered from poor samples and were limited by computational complexity of computing and backpropagating through higher order moments. The solution proposed by GANs to this problem was essentially to learn the distance function d() using another neural network titled the discriminator D(x). At first, it seems impossible to train the discriminator network D(x) using the ML framework, because we do not have access to the ground truth distribution p(x). But borrowing some preliminary concepts from game-theory, one can quickly realise that the objectives of the two networks, viz. Generator G and Discriminator D are opposite of each other. The Generator G is tasked to generate samples similar to ground-truth samples (by modelling the distribution p implicitly) and the Discriminator D is tasked to distinguish between the real and the generated samples (by modelling the distance function d() implicitly). This was the key insight that allowed Goodfellow et al. to propose an adversarial framework for successfully training generative models over very high-dimensional spaces. They showed in the paper through theoretical argumentation and through empirical experiments that, by defining the training as a mini-max game between two fairly capable networks, a nash-equilibrium can be achieved (refer fig. 2.6). At the nash-equilibrium, the Discriminator D is no longer able to distinguish between the real and the generated samples, while the Generator G generates samples completely indistinguishable from the real ones. This nashequilibrium thus yields the sought-after Generator model G which has implicitly learned to model the ground truth data distribution *p*.

Mathematically, the objective for the Discriminator is a binary-cross entropy between real and generated samples, defined as:

$$\mathcal{L}_{D} := \frac{1}{m} \sum_{i=0}^{m} -log(D(x^{i})) - log(1 - D(G(z^{i})))$$

While the loss for the Generator is to fool the Discriminator such that the Discriminator mis-classifies the generated samples as the real one.

$$\mathcal{L}_G := \frac{1}{m} \sum_{i=0}^m log(D(G(z^i)))$$



Figure 2.7: The forward and backward *markov chains* defined by Diffusion Models. The first step x_0 denotes the true data samples such as images while the final step x_T indicates pure Gaussian noise. All the intermediate x_t represent noisy versions of the data-samples.

We assume that the discriminator network outputs a valid normalized probability value between [0, 1]. The training of these networks proceeds as a mini-max game between the two networks where the Discriminator *minimizes* the objective \mathcal{L}_D and the Generator *maximizes* the objective \mathcal{L}_G during the training. In practice, the training loop first executes the Generator update-step (keeping the Discriminator constant) and then executes the Discriminator update-step (keeping the Generator constant).

2.3.2 Diffusion Models

Diffusion models are the state-of-the-art method for generative modelling at the time of writing this thesis. *Diffusion models* fall under the class of likelihood-based models which are trained to maximize the likelihood of the observed samples in the dataset and, similar to VAEs, can be used to compute the analytical likelihood of newly observed or doctored samples. Diffusion Models are the go-to choice for generative modelling problems due to their stable training objective and the ability to produce high-sample quality while training on billion-scale datasets.

As described in the figure 2.7, the concept of Diffusion Models is centered on the idea of defining a forward diffusion (noising) process $q(x_t|x_{t-1})$, for $t \in [0,T]$. The noising process converts the data samples into pure noise, i.e., $q(x_T) \approx q(x_T|x_{T-1}) =$

42

 $\mathcal{N}(0,I)$. The model then learns the reverse process $p(x_{t-1}|x_t)$, which iteratively converts the noise samples back into data samples starting from the purely Gaussian sample x_T . Given this formulation, it goes without saying that $p(x_0)$ and $q(x_0)$ both refer to the same thing, i.e. the ground truth data distribution whose observed samples are available in the dataset. We don't use $q(x_0)$ in the reverse markov chain defined by the $p(x_{t-1}|x_t)$ transitions to avoid confusion, and thus we may use these two interchangeably which can be delineated from the context.

The Denoising Diffusion Probabilistic Model (DDPM) [80], in particular, defines the noising transitions using a Gaussian distribution as

$$q(x_t|x_{t-1}) := \mathcal{N}(x_t; \sqrt{\alpha_t}x_{t-1}, (1-\alpha_t)I).$$

The sequence α_t defines a noise schedule for the diffusion process as:

$$\alpha_t = 1 - \beta_t, \quad \beta_t \in [0, 1], \quad \text{s.t. } \beta_t > \beta_{t-1} \forall t \in [0, T].$$

Although a linear schedule with T = 1000 is good for most cases, Dhariwal et al. [81] propose a consine schedule for the diffusion process which adds the noise more slowly compared to the linear schedule. Interestingly, the noisy samples x_t can be easily drawn from this distribution by recursively applying the reparameterization trick and using properties for addition and subtraction of Gaussian distributions:

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \varepsilon$$

with $\bar{\alpha}_t := \prod_{s=1}^t \alpha_s$ and $\varepsilon \sim \mathcal{N}(0, \mathbf{I})$

Please refer to eqns. (61-70) of the 'Understanding Diffusion Models' report by Luo [82], for a detailed derivation of this result. This is in fact the key mathematical insight that allows us to train Diffusion Models programmatically using ML frameworks. Having defined the forward process, DDPM defines the reverse denoising

process using another markov chain of Gaussian transitions:

$$p_{\theta}(x_{t-1}|x_t) := \mathcal{N}(x_{t-1}; \sqrt{\alpha_t} \mathcal{D}_{\theta}(x_t, t), (1 - \alpha_t) \mathbf{I}),$$

where, the \mathcal{D}_{θ} is the denoising network with learned parameters θ . Note that this reverse process is a learned component and is what allows to generate synthetic samples at test-time.

It is common to use the network $\mathcal{D}_{\theta}(x_t, t)$ to predict the noise component ε instead of the signal component x_{t-1} ; which has the interpretation of modelling the score of the marginal distribution $q(x_t)$ up to a scaled constant [80, 82]. However, the " x_0 -formulation" has recently been explored in the context of diffusion model distillation [83] and diffusion based generative modelling of text-conditioned videos [14]. Since this " x_0 formulation" is crucial for being able to train 3D diffusion models directly using only 2D supervision, in the subsequent chapters 6, 7, and 8, we detail the training and sampling procedure using this formulation as follows:

Training. Training the " x_0 -formulation" of a diffusion model \mathcal{D}_{θ} simply comprises of minimizing the following loss:

$$\mathcal{L} = \|\mathcal{D}_{\theta}(x_t, t) - x_0\|^2,$$

encouraging \mathcal{D}_{θ} to denoise sample $x_t \sim \mathcal{N}(\sqrt{\overline{\alpha}_t}x_0, (1-\overline{\alpha}_t)\mathbf{I})$ and predict the clean sample x_0 .

Sampling. Once the denoising network \mathcal{D}_{θ} is trained, sampling can be done by first starting with pure noise, i.e., $x_T \sim \mathcal{N}(0, \mathbf{I})$, and then iteratively refining it *T* times using the network \mathcal{D}_{θ} , which terminates with a sample from target data distribution $x_0 \sim q(x_0) = p(x)$:

$$x_{t-1} \sim \mathcal{N}(\sqrt{\bar{\alpha}_{t-1}}\mathcal{D}_{\theta}(x_t,t), (1-\bar{\alpha}_{t-1})I).$$

Lastly, note that this is a very focussed sampler of the literature available on Diffusion based generative models. Please refer to the comprehensive survey work by Calvin Luo [82] for more in-depth discussion on the mathematics behind the Diffusion Models.

In summary, I covered two of the most popular generative models which are also required for the subsequent chapters. These two models are specially unique because they setup two different paradigms for the task of generative modelling. More importantly, since Diffusion models have a stable learning objective, they are the preferred models right now for training massively large scale text-to-image and text-to-video models such as DALL-E [13], Stable-Diffusion [12], and Imagen-Video [14]. Although, works such as GigaGAN [15] are also pushing forward the state-of-the-art in GANs which competes with the Diffusion based state-of-the-art.

Chapter 3

Literature survey

In this chapter I try to cover the relevant related work on this topic. Note that this is a very diverse, crowded, and heavily competitive topic, so the coverage is far from being complete. Still, the attempt is to cover at least the main relevant works that are directly related to the proposed methods.

3.1 Discrete sample based representations

Computer vision and graphics have long experimented with different representations for working with visual data. While working with 2D images, they are ubiquitously represented as 2D grids of pixels; while due to the memory requirements, 3D models are often represented (and stored) in a sparse format, e.g., as meshes, sparse/hashed voxel grids, or as point clouds. In the context of images, since as early as the sixties [84], different ideas have been proposed to make pixels more expressive. One popular option is to store a fixed number (e.g., one) of zero-crossing for explicit edge boundary information [85, 86, 87, 88], by using curves [89], or augmenting pixels/voxels with more than one color [90, 91]. Another idea is to deform the underlying pixel grid by explicitly storing discontinuity information along general curves [92]. Loviscach [93] optimized MIP maps, such that the thresholded values match a reference. Similar ideas were also being explored for textures and shadow maps [94, 95], addressing specific challenges in sampling. In the 2D domain, the regular pixel grid format of images has proven to be amenable to machine learning algorithms because CNNs are able to naturally input and output regularly sampled

2D signals as pixel grids. As a result, these architectures can be easily extended to 3D to operate on voxel grids, and therefore can be trained for many learning-based tasks, e.g., using differentiable volume rendering as supervision [96, 97, 98, 99]. However, such methods are inefficient with respect to memory and are hence typically restricted to low spatial resolution.

3.2 Differentiable rendering.

Differentiable rendering [56] enables neural networks to be trained by losses on the resulting rendered images of a 3D representation, by allowing the gradient to be back-propagated through the network. This has shown to be a highly effective tool, especially for learning 3D representations that allow for novel view synthesis. Multiple works [100, 101, 102, 103, 104, 105] have focussed on designing neuralnetworks, either convolutional or otherwise, to go from spatial features to rendered pixels. Lately, methods that use volume tracing have been favored due to the advantage of flicker-free 2D rendering by design. NeRF [34] were the first to introduce this way of using differentiable rendering while using a neural 3D scene representation. Subsequently, many extensions of this have been proposed, to increase quality, robustness to scene type, and rendering speed [34, 106, 107, 108, 109, 110, 111, 112, 113, 114]. Although using an MLP to learn a continuous scene representation has been shown to be able to lead to very high quality view synthesis results, such MLPs are not amenable in the generative contexts as ours.

3.3 Learned neural representations

Recently, coordinate-based MLPs representing continuous signals have been shown to be able to dramatically increase the representation quality of 3D objects [115] or reconstruction quality of 3D scenes [32, 34]. However, such methods incur a high computational cost, as the MLP has to be evaluated, often multiple times, for each output signal location (e.g., pixel) when performing differentiable volume rendering [34, 116, 117, 118]. In addition, this representation is not well suited for post-training manipulations as the weights of the MLP have a global effect on the structure of the scene. To fix the slow execution, sometimes grid-like representations

are fit post-hoc to a trained NeRF model [113, 112, 111, 110], however such methods are unable to reconstruct scenes from scratch. As a result, there has been an interest in hybrid methods that store learned features in spatial data structures, and accompany this with an MLP, often much smaller, for decoding the interpolated neural feature signal at continuous locations. Examples of such methods store learned features on regular grids [99, 119], sparse voxels [120, 121], point clouds [102], local crops of 3D grids [122], or on intersecting axis-aligned planes (Triplanes) [67]. Investigating representations suitable for efficiently representing complex signals is an active area of research. Reporting a finding similar to ours in ReLU-Fields (chapter 4), DVGo [65] proposes the use of a "post-activated" (i.e., after interpolation) density grid for modelling high-frequency geometries. They model the view-dependent appearance through a learned feature grid which is decoded using an MLP. They in-fact show comprehensive experimental evaluation, on multiple datasets comparing to multiple baselines, for the task of image-based 3D scene reconstruction. Plenoxels [123] proposes the use of sparse grid structure for modeling the scene with ReLU activation and, similar to our experiments, also uses spherical harmonic coefficients [111] for modeling view-dependent appearance. Instant-NGP [66] proposes a hierarchical voxel-hashing scheme to store learned features and uses a small MLP decoder for converting them into geometry and appearance. Their reconstruction-times are significantly lower than the others because of their impressively engineered GPU implementation.

3.4 2D-to-3D Encoding

Most of the 3D encoding mechanisms proposed till now have been in the context of a larger problem such as MVS or NVS, and there has also not been a standalone principled study of the encoding mechanisms yet. Nevertheless, the early works [124] constructed pixel-disparity based 3D cost-volumes for MVS (Multi-View Stereo) problems. These raw-pixel based cost-volumes were soon superseded by 2D deep image feature based ones [125, 126, 105, 70, 71] due their memory-compactness and information expressiveness. These approaches typically proceed as: first obtain per-image-per-pixel features using large pretrained image networks such as ResNet [43] or DinoV2 [127]; then un-project these features into the 3D space using the camera parameters associated with the views. The un-projected features are then accumulated into the feature-voxel grid yielding the cost-volume to be used in various 3D problem contexts. Apart from voxel-grids, recent works also splat these features either on Triplanes [128], or on the surface of proxy meshes. The most similar work to our proposed GOEmbed method (chapter 8) in terms of the idea is the one from Bond-Taylor and Willcocks titled GON (Gradient Origin Networks) [129], although their proposal has no context of 3D encodings. We take inspiration from GONs, but our proposed GOEmbeddings are different from them in that: (i) while the purpose of GON is to obtain a compressed latent space, our GOEmbed is aimed at obtaining a coarse partial estimate of the plenoptic 3D scene given 2D image observations; and (ii) our GOEmbed encodings are much more local compared to the latent embeddings obtained using the GON.

3.5 2D generative models.

Generative modeling for image synthesis learns a distribution of colour values over the pixels of an image, and has seen tremendous progress recently, with GANs and Diffusion models being the de-facto standard for synthesizing realistic looking images [130, 131, 132, 133, 134]. Recent works [135] further improve result quality in data-sparse regimes, while others [136] apply signal processing techniques to the generator architecture to make the mapping from latent space (noise-vectors) to the image-space (pixels) as smooth and alias-free as possible. Thereby, allowing for realistic looking latent-space interpolations that can pass for real videos up to some physics breaking constraints. Other contenders for generative modelling include VAEs [137, 138, 139, 140], flow-based models [141, 142, 143], noise-diffusion based models [144, 145, 146, 147], and even hybrids of these methods [148, 147, 149, 150]. Key to training such methods is the availability of large scale image collections, and often times such works are used on specific target domains, such as portrait photos. Single Image GANs (SinGANs) [151, 152] learn to generate the distribution of *patches* of a single image, in a progressive coarse-to-fine manner so as to generate plausible variations from that one single image. Hinz et al. [152] followed SinGAN with tips and tricks that improve the training and generated quality of single image GANs. Such approaches avoid the problem of needing a large dataset, while still enabling useful applications such as retargetting. However, they are restricted to repeated, or stochastic-like patch-based variations. In spite of all this progress in the field of the 2D generative modelling, these models still lack 3D inductive bias and seemingly simple tasks such as camera view transform are not possible with these 2D generative models.

3.6 3D generative models.

Due to the lack of large scale real-world datasets, much of the research in 3D generative modeling has stayed in the synthetic realm such as modeling only 3D shapes [153, 154, 155, 156, 157, 158], or materials [159]. Other methods use synthetic datasets for predicting scene structure and use differentiable rendering for training these models end-to-end [160, 161, 162, 163]. Recently, methods that directly model 3D scenes, either using explicit or implicit neural scene representations, have been gaining popularity [97, 98, 116, 117, 118, 164, 165, 166]. Another successful line of works [98, 118] uses a neural renderer to render features from a volumetric grid, followed by per-image 2D CNNs used for upsampling, and as such are not multiview consistent. In contrast, methods such as PiGAN and GRAF [116, 117, 164] are view-consistent by design since they use an implicit neural representation and explicit-implicit neural representation respectively for the 3D structure, using a physically based rendering equation for obtaining the final 2D images. Works such as CIPS and StyleNeRF [165, 166] produce impressive results performing 3D synthesis with multi-view consistency. But their GAN setups also only use 2D discriminators, and show results on some of the easier domains such as faces. Such approaches are designed to be trained only through 2D (image) supervision, and work best in cases where large datasets can be obtained for limited domains, such as faces or cars. Generative Adversarial Learning (GAN) [78] learns a generator network so that its "fake" samples cannot be distinguished from real images by a second discriminator network. Approaches such as PlatonicGAN [97], HoloGAN [98], and PrGAN [167] introduced 3D structure into the generator network, achieving 3D shape generation with only image-level supervision. Our proposed method of HoloDiffusion (chapter 6) is related to those as it renders images from a generated voxel grid, as well as to HoloGAN [98], which renders features and then converts them into an image by a lightweight 2D convolutional network. Other voxel-based 3D generators include VoxGRAF [168] and NeuralVolumes [63].

More recently, 3D generators have built on neural radiance fields [34]. GRAF [117] was the first to adopt the NeRF framework; analogous to PlatonicGAN, they generate the parameters of an MLP which renders realistic images of the object from a random viewpoint. This idea has been improved in StyleNeRF [166] and EG3D [169] by adding a 2D convolutional post-processing step after emission-absorption rendering, which is analogous to our super-resolution network of HoloFusion [71] (chapter 7). EG3D also introduced a novel 'tri-plane' representation of the radiance field which, in a memory efficient manner, factorises the latter into a triplet of 2D feature planes. EG3D inspired several improvements such as GAUDI [170] and EpiGRAF [171]. Mesh-based 3D generators have been explored in Wu et al. [153]. Recently, GET3D [172] replaced the radiance field with a signed distance function to regularise the representation of geometry. The latter is converted into a mesh and rendered in a differentiable manner by using the marching tetrahedral representation [173].

Modeling 3D with diffusion. Diffusion methods [80] have recently became the go-to framework for generative modeling of any kind, including 3D generative modeling. The first applications of diffusion to 3D considered point-cloud generators trained on synthetic data [174, 175, 176].

3D distillation of 2D diffusion models. More recently, DreamFusion [26] ported the idea of distillation to diffusion models: they extract a neural radiance field so that its renders match the belief of a pre-trained 2D diffusion generator [177, 81, 12]. They introduce the Score Distillation Sampling (SDS) loss which makes distillation

51

relatively efficient (but still in the order of several minutes for a single 3D sample). Their generation can be conditioned by an image or by a textual description, making the process rather flexible. Magic3D [178] further increases the quality of the output by distilling a mesh-based 3D representation instead of a radiance field.

Image-conditioned 3D. The idea of distillation has been applied to few-view conditioned reconstruction in [179, 180, 181, 182, 183]. SparseFusion [182] employs a 3D-based new-view synthesis model followed by a 2D diffusion upsampler. They complete the process by 3D distillation, ensuring that the generated views of the object are consistent. NeRFDiff [180] and 3DiM [179] bypass an explicit 3D model and directly generate new views of an object using a 2D image generator and, in the case of NeRFDiff, refine the results using distillation.

While SparseFusion and NeRFDiff need to be trained on a dataset of objectcentric multi-view images with pose information, RealFusion [181] and NeRDi [183] can be used for zero-shot monocular 3D reconstruction, starting from a pre-trained 2D diffusion model. Given a single image as input, they automatically generate a prompt for the diffusion model, using a form of prompt inversion, and then use distillation to extract a radiance field.

Unconditional generation. Most relevant to us, unconditional generation, i.e., generation which does not require either text or image conditioning, was explored in RODIN and DiffRF [184, 185]. While RODIN and DiffRF [185, 184] train generators given synthetic 3D ground truth, similar to us, Our proposed HoloDiffusion [70] (chapter 6) is supervised only with real object-centric images and camera poses. While HoloDiffusion was the first to demonstrate successful training on real image data, its renders contain considerably lower amount of detail than samples from a conventional 2D image generator that uses diffusion. We thus leverage a 2D diffusion upsampler, conditioned on the lower-fidelity HoloDiffusion renders, to distill higher resolution images and, eventually, 3D models in our proposed extension titled HoloFusion [71] (chapter 7).

Chapter 4

ReLU-Fields: The Little Non-linearity That Could

4.1 Background and Contributions

I had experience of working on Generative Models and Deep Learning prior to starting the PhD., for instance the MSG-GAN publication [134], but I was quite new to the world of 3D applications. This project was an initial exploration in Neural Radiance Fields and a chance for me to understand the basics of 3D applications, online and offline forms of rendering, etc. We began with our NeRF exploration and decided to try out a version of SinGAN [151] in 3D. The idea of ReLU-Fields was essentially an accidental discovery by Tobias when I had first implemented volumetric-voxel based Radiance field optimization pipeline. He uncovered that, in my implementation, I had applied ReLU on top of the interpolated voxel-grids thinking that the ReLU function is a transfer function. Which was clarified to me later that transfer functions are applied prior to interpolation and are basically a means of tone-mapping various vector data into colour information that can be rendered and visualized. After this discovery, we performed an initial set of experiments where we found that the obtained sharpness of fitted ReLU-fields was much higher than traditional linearly-interpolated voxel grids. We (Niloy, Tobias, Oliver and I) found this insight interesting, in that, it pinpointed exactly what allows NeRFs to obtain high-quality reconstructions; which in this case, is a simple ReLU non-linearity. It



Figure 4.1: We present a method to represent complex signals such as images or 3D scenes, both volumetric (left) and surface (right), on regularly sampled grid vertices. Our method is able to match the expressiveness of coordinate-based MLPs while retaining reconstruction and rendering speed of voxel grids, *without* requiring any neural networks or sparse data structures.

was realised that this could be a useful insight for the research community and hence we decided to go forward with the publication. I worked on the implementation and experimentation entirely for this project and also majorly contributed to the write-up and dissemination of the publication.

4.2 Introduction

In many recent works, multi-layer perceptions (MLPs) have been shown to be suitable for modeling complex spatially-varying functions including images and 3D scenes. Although the MLPs are able to represent complex scenes with unprecedented quality and memory footprint, this expressive power of the MLPs, however, comes at the cost of long training and inference times. On the other hand, bilinear/trilinear interpolation on regular grid-based representations can give fast training and inference times, but cannot match the quality of MLPs without requiring significant additional memory. Hence, in this work, we investigate what is the *smallest* change to grid-based representations that allows for retaining the high fidelity result of MLPs while enabling fast reconstruction and rendering times. We introduce a surprisingly simple change that achieves this task – *simply allowing a fixed nonlinearity (ReLU) on interpolated grid values.* When combined with coarse-to-fine optimization, we show that such an approach becomes competitive with the stateof-the-art. We report results on radiance fields, and occupancy fields, and compare against multiple existing alternatives. Code and data for the project are available at https://geometry.cs.ucl.ac.uk/projects/2022/relu_fields.

Coordinate-based MLP have been shown to be capable of representing complex signals with high fidelity and a low memory footprint. Exemplar applications include NeRF [34], which encodes lighting-baked volumetric radiance-density field into a single MLP using posed images; LIFF [186], which encodes 2D image signal into a single MLP using multi-resolution pixel data. Alternatively, a 3D shape can be encoded as an occupancy field [32, 187], and a signed distance field [188].

A significant drawback of such approaches is that MLPs are both slow to train and slow to evaluate, especially for applications that require multiple evaluations per signal-sample (e.g. multiple per-pixel evaluations during volume tracing in NeRFs). On the other hand, traditional data structures like *n*-dimensional grids are fast to optimize and evaluate, but require a significant amount of memory to represent high frequency content (see Figure 4.4). As a result, there has been an explosion of interest in hybrid representations that combine fast-to-evaluate data structures with coordinate-based MLPs, e.g., by encoding latent features in regular [65] and adaptive [120, 102, 121, 66] grids and decoding linearly interpolated "neural" features with a small MLP.

In this project, we revisit regular grid-based models and look for the *minimum* change needed to make such grids perform on par with "neural" representations. As the key takeaway message, we find that simply using a ReLU non-linearity on top of interpolated grid values, without any additional learned parameters, optimized in a progressive manner already does a surprisingly good job, with minimal added complexity. For example, in Figure 4.1 we show results in the context of representing volumes (left) and surfaces (right) and on regularly sampled grid vertices respectively. As additional benefits, these grid based 3D-models are amenable to generative modeling, and to local manipulation.

In summary, we present the following contributions:

- we propose a minimal extension to grid-based signal representations, which we refer to as ReLU-Fields
- 2. we show that this representation is *simple*, does not require any neural net-



Figure 4.2: Representing a ground-truth function (**blue**) in a 1D (a) and 2D (b) grid cell using the linear basis (**yellow**) and a ReLU-Fields (**pink**). The reference has a c1-discontinuity inside the domain that a linear basis cannot capture. A ReLU-Field will pick two values y_1 and y_2 , such that their interpolation, after clamping will match the sharp c1-discontinuity in the ground-truth (**blue**) function.

works, is directly *differentiable* (and hence easy to optimize), and is fast to *optimize and evaluate* (i.e. render)

3. we empirically validate our claims by showing applications where ReLU-Fields plug in naturally: first, image-based 3D scene reconstruction; and second, implicit modeling of 3D geometries.

4.3 It's just a little ReLU

We look for a representation of *n*-valued signals on an *m*-dimensional coordinate domain \mathbb{R}^m . For simplicity, we explain the method for m = 3. Our representation is strikingly simple. We consider a regular (m = 3)-dimensional $(r \times r \times r)$ -grid *G* composed of *r* voxels along each side. Each voxel has a certain size defined by its diagonal norm in the (m = 3)-dimensional space and holds an *n*-dimensional vector at each of its $(2^{m=3} = 8)$ vertices. Importantly, even though they have matching number of dimensions, these values do not have a direct physical interpretation (e.g., color, density, or occupancy), which always have some explicitly-defined range, e.g., [0, 1] or $[0, +\infty)$. Rather, we store unbounded values on the grid; and thus for technical correctness, we call these grids "feature"-grids instead of signal-grids. The features at grid vertices are then interpolated using (m = 3)-linear interpolation, and

followed by a *single non-linearity*: the ReLU, i.e., function $\text{ReLU}(x) = \max(0, x)$ which maps negative input values to 0 and all other values to themselves. Note that this approach does not have any MLP or other neural-network that interprets the features, instead they are simply clipped before rendering. Intuitively, during optimization, these feature-values at the vertices can go up or down such that the ReLU clipping plane best aligns with the c1-discontinuities within the ground-truth signal. Figure 4.2 illustrates this concept.

As a didactic example, we fit an image into a 2D ReLU-Field grid similar to [35], where grid values are stored as floats in the $(-\infty, +\infty)$ range. For any query position, we interpolate the grid values before passing through the ReLU function (see Algorithm 3). Since the image-signal values are expected to be in the [0,1] range, we apply a hard-upper-clip on the interpolated values just after applying the ReLU. We can see in figure 4.3 that ReLU-field allows us to represent sharp edges at a higher fidelity than bilinear interpolation (without the ReLU) at the same resolution grid size. One limitation of this representation is that it can only well represent signals that have sparse c1-discontinuities, such as this flat-shaded images and as we show later, 3D volumetric density. However, other types of signals, such as natural images, do not benefit from using a ReLU-Fields representation (see supplementary video at https://www.youtube.com/watch?v=GcRgqzWh4FA).

Algorithm 3 Fetching a 2D ReLU field.

```
1: procedure RELUFIELD2D(G, x)
```

- 2: $\mathbf{x}_g := FLOOR(\mathbf{x})$
- 3: $\mathbf{x}_{\mathbf{f}} := FRAC(\mathbf{x})$
- 4: $y_{00} := \text{FETCH}(G, \mathbf{x}_{g} + (0,0))$
- 5: $y_{01} := \text{FETCH}(G, \mathbf{x}_{g} + (0, 1))$
- 6: $y_{10} := \text{FETCH}(G, \mathbf{x}_{g} + (1,0))$
- 7: $y_{11} := \text{FETCH}(G, \mathbf{x}_g + (1, 1))$
- 8: $y := BILINEAR(y_{00}, y_{01}, y_{10}, y_{11}, \mathbf{x}_{f})$
- 9: **return** RELU(y)
- 10: end procedure



Figure 4.3: Representing an image with a standard pixel grid bi-linearly interpolated to a larger size (Grid) versus a ReLU-Field of the same size (ReLUField). The grid-size of the variants, ReLUField and Grid, is 64x smaller; while of, ReLUFieldL and GridL, is 32x smaller than the source image-resolution *along each dimension*. Note that the 'L' variants have a bigger grid-size and hence less smaller than the GT raster image. Simply adding a ReLU allows for significantly more sharpness and detail to be expressed. Hence, we can say that the humble ReLU is truly *the little non-linearity that could*.

4.4 Applications

We now demonstrate two different applications of ReLU-Fields; NeRF-like 3D scene-reconstruction (4.4.1), and 3D object reconstruction via occupancy fields

4.4. Applications



Figure 4.4: Qualitative comparison between NeRF-PT, GridL and ReLUFieldL. Grid-based versions converge much faster, and we can see significant sharpness improvements of ReLUFieldL over GridL, for example in the leaves of the plant. See also supplementary video.

(4.4.2).

4.4.1 Radiance Fields

In this application, we discuss how ReLU-Field can be used in place of the coordinatebased MLP in NeRF [34]. Input to this algorithm are a set of images $\mathcal{I} = \{I_1, \ldots, I_n\}$ and corresponding camera poses $\mathcal{C} = \{C_1, \ldots, C_n\}$, where each camera pose consists of $C = \{R, T, H, W, F\}$; *R* is the rotation-matrix ($R \in \mathbb{R}^{3 \times 3}$), *T* is the translation-vector ($T \in \mathbb{R}^3$), *H*, *W* are the scalars representing the height and width respectively, and *F* denotes the focal length of the assumed simple pin-hole camera. We assume that the respective poses for the images are known either through hardware calibration or by using structure-from-motion [75].

We denote the rendering operation to convert the 3D scene representation \mathcal{R} and the camera pose C into an image as $\zeta(\mathcal{R}, C)$. Thus, given the input set of images \mathcal{I} and their corresponding camera poses \mathcal{C} , the problem is to recover the underlying 3D scene representation \mathcal{R} such that when rendered from any $C_i \in \mathcal{C}, \mathcal{R}$ produces rendered image \hat{I}_i as close as possible to the input image I_i , and produces spatio-temporally consistent \hat{I}_j for poses $C_j \notin \mathcal{C}$.

Scene representation We model the underlying 3D scene representation \mathcal{R} , which is to be recovered, by a ReLU-Field. The vertices of the grid store, first, raw pre-relu density values in $(-\infty,\infty)$ that model geometry, and, second, the second-degree SH coefficients [111, 189] that model view-dependent appearance. The relu is only applied to pre-relu density, not to appearance.

We directly optimize values at the vertices to minimize the photometric loss between the rendered images \hat{I} and the input images I. The optimized grid G^* , corresponding to the recovered 3D scene \mathcal{R} , is obtained as:

$$G^* = \arg\min_{G} \sum_{i=1}^{n} \|I_i - \overbrace{\zeta(G, C_i)}^{I_i}\|_2^2.$$
(4.1)

Implementation details Similar to NeRF, we use the EA (emission-absorption) raymarching model [72, 97, 34] for realizing the rendering function ζ . The grid is scaled to a single global AABB (Axis-Aligned-Bounding-Box) that is encompassed by the camera frustums of all the available poses C, and is initialized with uniform random values. We optimize the vertex values using Adam [190] with a learning rate of 0.03, and all other default values, for all examples shown.

We perform the optimization progressively in a coarse-to-fine manner similar to [191]. Initially, the feature grid is optimized at a resolution where each dimension

Table 4.1: Evaluation results on 3D synthetic scenes. Metrics used are PSNR (↑) / LPIPS (↓). The column NeRF-TF* quotes PSNR values from prior work [34], and as such we do not have a comparable runtime for this method.

Scene	NeRF-TF*		NeRF-PT		Grid		GridL		ReLUField		ReLUFieldL		RFLong		RFNoPro	
	PSNR	LPIPS	PSNR	LPIPS	PSNR	LPIPS	PSNR	LPIPS	PSNR	LPIPS	PSNR	LPIPS	PSNR	LPIPS	PSNR	LPIPS
Chair	33.00	0.04	33.75	0.03	25.53	0.12	27.08	0.11	31.50	0.05	32.39	0.03	31.77	0.05	13.85	0.48
Drums	25.01	<u>0.09</u>	23.82	0.12	19.85	0.20	20.70	0.17	23.13	0.09	25.15	0.06	23.78	0.09	10.74	0.52
Ficus	30.13	0.04	28.96	0.04	22.10	0.13	23.61	0.11	25.89	0.06	27.37	<u>0.04</u>	26.11	0.05	13.21	0.47
Hotdog	36.18	0.12	33.52	<u>0.06</u>	28.53	0.12	29.83	0.10	34.65	0.03	<u>35.72</u>	0.03	34.70	0.03	12.22	0.53
Lego	32.54	0.05	28.36	0.08	23.76	0.17	23.97	0.15	28.83	0.06	30.78	0.03	29.64	0.05	10.63	0.56
Materials	29.62	0.06	29.23	0.04	21.87	0.18	22.74	0.13	27.41	0.06	28.23	0.05	28.23	0.05	8.99	0.55
Mic	<u>32.91</u>	0.02	33.08	0.02	25.87	0.08	25.91	0.08	31.88	0.03	32.62	0.02	31.22	0.03	12.47	0.41
Ship	<u>28.65</u>	0.20	29.22	0.14	23.86	0.25	22.54	0.24	26.86	0.14	28.02	0.12	27.39	0.13	9.92	0.59
Average	31.01	0.07	29.99	0.07	23.92	0.16	24.54	0.14	28.77	0.07	<u>30.04</u>	0.05	29.10	<u>0.06</u>	11.50	0.51
Time (recon)	-	_	11h:2	1 m:00s	00h:03	3m:41s	00h:1	0m:02s	00h:03	3m:41s	00h:10)m:36s	10h:5	1m:29s	00h:07	m:11s
Time (render)	-	-	16,36	3.0 ms	9.0	ms	99.	1 ms	9.1	ms	99.5	5 ms	9.2	1 ms	9.8	ms

is reduced by a factor of 2^4 . After a fixed number of iterations at each stage N, the grid resolution is doubled and the features on the feature-grid G are tri-linearly upsampled to initialize the next stage. This proceeds until the final target resolution is reached.

Evaluation We perform experiments on the eight synthetic Blender scenes used by NeRF [34], viz. Chair, Drums, Ficus, Hotdog, Lego, Materials, Mic, and Ship and compare our method to prior works, baselines, and ablations. We also show an extension of ReLU-Fields to one of their real world captured scenes, named Flowers.

First, we compare to the mlp-based baseline NeRF [34]. For the purpose of these experiments though, we use the public nerf-pytorch version [192] for comparable training-time comparisons since all our implementations are in PyTorch. For disambiguation, we refer to this PyTorch version as NeRF-PT and the original one as NeRF-TF and report scores for both. Second, we compare to two versions of traditional grids where vertices store scalar density and second-degree SH approximations of the appearance, namely Grid (i.e., 128³ grid) and GridL (i.e., 256³ grid). Finally, we compare to our approach at the same two resolutions, ReLUField and Re-LUFieldL. The above four methods are optimized with the same progressive growing setting with N = 2000, and all the same hyperparameters except the grid resolution. We report PSNR and LPIPS [193] computed on a held-out test-set of Image-Pose pairs different from the training-set (\mathcal{I} , \mathcal{C}). All training times were recorded on 32GB-V100 GPU while the inference times were computed on RTX 2070 Super.

Our method is implemented entirely in PyTorch and does not make use of any custom GPU kernels.

Table 4.1 summarizes results from these experiments. We can see that traditional physically-based grid baselines Grid and GridL perform the worst, while our method has comparable performance to NeRF-PT and is much faster to reconstruct and render. This retains the utility of grid-based models for real-time applications without compromising on quality. Figure 4.4 demonstrates qualitative results from these experiments.

Ablations We ablate the components described in 4.4.1, and and also include the results in Table 4.1 in the last two columns. RFLong is a normal ReLU-Field optmized for a much longer time (comparable to NeRF-PT's training time). We see minor improvement over the default settings, however we can see that the optimization time plays less of a role than the resolution of the grid itself (ReLUFieldL outperforms RFLong). RFNoPro is trained without progressive growing for the same number of total steps. We see that it yields a much lower reconstruction quality, indicating that progressive growing is critical for the grid to converge to a good reconstruction.

Real scene extension Similar to the real-captured-360 scenes from the NeRF, we also show an extension of ReLU-Fields to modeling real scenes. In this example, we model the background using a "MultiSphereGrid" representation, as proposed by [194]. Please note that the background grid is modeled as a regular bilinear grid without any ReLU. For simplicity, we use an Equi-rectangular projection (ERP) instead of Omni-directional stereo (ODS) for mapping the Image-plane to the set of background spherical shells. Fig. 4.5 shows qualitative results for this extension after one hour of optimization. Here, we can see that the grid does a good job of representing the complex details in the flower, while the background is modeled reasonably well by the shells.

4.4.2 Occupancy Fields

Another application of coordinate-based MLP is as a representation of (watertight) 3D geometry. Here, we fit a high resolution ground-truth mesh, as a 3D occupancy field [32] into a ReLU-Field. One might want to do this in order to, for example, take



Figure 4.5: Qualitative results for the real-captured scene extension of ReLU-Fields on Flowers. We decompose the scene into a series of spherical-background shells and a foreground ReLU-Field layer, which are alpha-composited together to give final novel view renderings. The top-left visualization shows the composite of the background spherical shells un-projected onto a 2D image-plane.

advantage of the volumetric-grid structure to learn priors over geometry, something that is harder to do with meshes or coordinate-based MLP directly.

Occupancy representation The core ReLU-Field representation used for this application only differs from the radiance fields setup (see 4.4.1) as follows: First, since we are only interested in geometry, we do not store any SH coefficients on the grid, and simply model volumetric occupancy as a probability from [0, 1]. Second, as supervision, we use ground truth point-wise occupancy values in 3D (i.e., 1, if the point lies inside the mesh, and 0 otherwise), rather than rendering an image and applying the loss on the rendered image. Finally, since the ground truth occupancy values are binary, we use a binary cross entropy (BCE) loss. Thus, we obtain the optimized grid G^* as,

$$G^* := \arg\min_{G} \sum_{\mathbf{x} \in \mathcal{B}} BCE(O(\mathbf{x}), RELUFIELD3D(tanh(G), \mathbf{x}))$$
(4.2)

where, O is the ground truth occupancy, **x** denote sample locations inside an axisaligned bounding box \mathcal{B} , BCE denotes the binary cross entropy loss, and G represents the ReLU-Field grid. Note that we use the tanh to limit the grid values in (-1, 1), although other bounding functions, or tensor-normalizations can be used.

Implementation details We initialize the grid with uniform random values. The supervision signal comes from sampling random points inside and around the tight AABB of the GT high resolution mesh, and generating the occupancy values for those points by doing an inside-outside test on the fly during training. For rendering, we directly show the depth rendering of the obtained occupancy values. We define the grid-extent and the voxel size by obtaining the *AABB* ensuring a tight fit around the GT mesh.

Table 4.2: Evaluation results on modeling 3D geometries as occupancy fields. Metric usedis Volumetric-IoU [32]. The baseline MLP is our implementation of OccupancyNetworks [32].

	MLP	Grid	ReLUField
Thai Statue	<u>0.867</u>	0.827	0.901
Lucy	0.920	0.883	0.935
Bimba	<u>0.983</u>	0.978	0.987
Grog	<u>0.961</u>	0.947	0.971
Lion	0.956	<u>0.970</u>	0.979
Ramses	<u>0.973</u>	0.961	0.978
Dragon	<u>0.886</u>	0.761	0.896
Average volumetric-IoU	<u>0.935</u>	0.903	0.949

Evaluation Figure 4.6 shows the qualitative results of the different representations used for this task. We can see that a ReLUField in this case yields higher quality reconstructions than a standard Grid, or a coordinate-based MLP. Quantitative scores, Volumetric-IoU as used in [32], for the ThaiStatue, Lucy, Bimba, Grog, Lion, Ramses, and Dragon models are summarized in 4.2. ReLU-Field and Grid require 15 mins, while MLP requires 1.5 hours for training.

4.5 Limitations

Our approach has some limitations. First, the resulting representations are large. A ReLU-Field of size 128³ used for radiance fields (i.e., with SH coefficients) takes 260Mb, and the large version at 256³ takes 2.0 Gb of storage. We believe that



Figure 4.6: Qualitative results for the occupancy fields comparing Grid, MLP, and ReLU-Field.

combining ReLU-Field with a sparse data structure would see significant gains in performance and reduction in the memory footprint. However, in this work we emphasize the simplicity of our approach and show that the single non-linearity alone is responsible for a surprising degree of quality improvement.

ReLU-Field also cannot model more than one "crease" (i.e., discontinuity) per grid cell. While learned features allow for more complex signals to be represented, they do so at the expense of high compute costs. The purpose of this work is to refocus attention on *what is actually required for high fidelity scene reconstruction*. We believe that the task definition and data are responsible for the high quality results we are seeing now, and show that traditional approaches can yield good results with minor modifications, and neural networks may not be required. However, this is just one data-point in the space of possible representations, for a given specific task we expect that the optimal representation may be a combination of learned features, neural networks, and discrete signal representations.

4.6 Summary

In summary, we presented ReLU-Field, an almost embarrassingly simple approach for representing signals; *storing unbounded data on N-dimensional grid, and apply-*

4.6. Summary

ing a single ReLU after linear interpolation. This change can be incorporated at virtually no computational cost or complexity on top of existing grid-based methods, and strictly improve their representational capability. Our approach does not rely on any learned parameters, special initialization, or neural networks; and performs comparably with state-of-the-art approaches in only a fraction of the time.

Chapter 5

3inGAN: Learning a 3D Generative Model from Images of a Self-similar Scene

5.1 Background and Contributions

As alluded to earlier, one of the challenges towards achieving 3D generative models is lack of sufficient-quality and large-scale 3D datasets. During the timeline of this project, this challenge was particularly significant since datasets such as Co3D [76] and OmniObject3D [195] were not released. Thus, being equipped with the basics of 3D volumetric representations and differentiable rendering from the previous project, we decided to get started with 3D generative modelling by temporarily narrowing down the problem statement further to patch-based 3D generative modelling. Inspired by the seminal work of SinGAN [151] (Shaham et al), we narrowed down our problem statement to: "Can we train a 3D generative model using only a single 3D scene?". This project is essentially a journey into finding an answer to this question. It turns out that a naive translation of 2D SinGAN to 3D is not enough, and we detail these challenges and how to overcome them in this chapter.

In terms of contributions, I worked on the implementation and experimentation entirely for this project and also majorly contributed to the write-up and dissemination of the paper. The supervisors Niloy, Tobias, and Oliver helped with critical analysis of the results, drafting of the paper and illustrations of ideas. From this project, I got to learn a lot about how scientific figures should be illustrated to get the idea across clearly and coherently from Tobais. I try to apply these learnings in the subsequent chapter 8.

5.2 Introduction

We introduce 3INGAN, an unconditional 3D generative model trained from 2D images of a single self-similar 3D scene. Such a model can be used to produce 3D "remixes" of a given scene, by mapping spatial latent codes into a 3D volumetric representation, which can subsequently be rendered from arbitrary views using physically based volume rendering. By construction, the generated scenes remain view-consistent across arbitrary camera configurations, without any flickering or spatio-temporal artifacts. During training, we employ a combination of 2D, obtained through differentiable volume tracing, and 3D GAN losses, across multiple scales, enforcing realism on both its 2D renderings and its 3D structure. We show results on semi-stochastic scenes of varying scale and complexity, obtained from real and synthetic sources. We demonstrate, for the first time, the feasibility of learning plausible view-consistent 3D scene variations from a single exemplar scene and provide qualitative and quantitative comparisons against two recent related methods. Code and data for the project are available at https://geometry.cs.ucl.ac.uk/group_website/projects/2022/3inGAN.

In the context of images, unconditional generative models, such as GAN, learn to map latent spaces to diverse yet realistic high resolution images – notable architectures include StyleGan [131] and BigGAN [133]. Furthermore, these models have been shown to contain high-level semantics in their latent space mappings, allowing powerful post-hoc image editing operations, such as changing the appearance and expression of a generated person [196, 197, 198]. One key question therefore, is whether it is possible to learn similar generative models for *3D* scenes.

There are, however, two key challenges. First, 3D generation suffers from *data scarcity* as obtaining large and diverse datasets for 3D data, both geometry and



Figure 5.1: Single scene 3D remixes. We introduce 3INGAN that takes a set of 2D photos of a *single* self-similar scene to produce a generative model of 3D scene remixes, each of which can be rendered from arbitrary camera configurations, without any flickering or spatio-temporal artifacts. Bottom row insets show zooms from different generative samples, rendered from the same camera view, to highlight the quality and diversity of the results.

appearance, is significantly more challenging than for 2D data, where one can simply scrape images from the internet. Second, generative models struggle as the *domain complexity* increases, as well as when datasets are not pre-aligned. This problem is even more severe in 3D, due to the added scene complexity, both in terms of scene

structure and object appearance, and the fact that 3D models and scans often come with their own coordinate systems and/or scaling.

In this work, we propose 3INGAN, a solution to address both problems in a restrictive setting: an unconditional generative model for 3D scenes that works on a per-scene basis for self-similar configurations. As our method does not require a large 3D dataset during training, it could be used across a wide range of real world domains. By restricting the data domain to a single 3D scene, we simplify the unconditional generation problem space to one with limited domain complexity, allowing us to learn a high-quality generator. We were inspired by similar approaches proposed for 2D images, e.g., SinGAN [151]. However, extending such approaches to 3D is nontrivial, as in 3D, one must be able to generate arbitrary views in a way that is multi-view consistent. We handle this by directly generating a 3D scene representation that is, by definition, multiview consistent. In particular, we generate a regularly sampled nonlinearly-interpolated-voxel grid [64] due to its simplicity, local (feature) influence, rendering efficiency, and its amenability to the prevailing convolutional generator and discriminator architectures in 2D/3D. In order to achieve realism, both in terms of 3D structure and 2D image appearance, we simultaneously use 3D feature patches and 2D image patches to obtain gradients from the 3D and 2D discriminators, respectively. We link the 3D and 2D domains by using a differentiable volume rendering module to interpret the feature grid as RGB images.

Note that recent neural 3D generators (e.g., BlockGAN [199], GRAF [117], π -GAN [116]) are trained on image/shape collections, and are not easily applicable in our setup (see sec. 5.4 for comparison). For example, Fig. 5.3 shows a sampling of "remixed" results generated from images of a school of fish.

In summary, ours is the first work to introduce an unconditional generative model from a *single 3D scene*. In particular, we investigate scenes with some degree of stochastic structure, which are suitable to shuffling or "remixing" the scene content into a new 3D scene that makes sense. In addition, we make a further simplifying assumption in that we drop the view specific effects and reconstruct Lambertian scenes. We evaluate our proposed method on a series of synthetic and real scenes

and show that our approach outperforms baselines in terms of quality and diversity.

5.3 **3inGAN** approach

Motivation. As input, we require an exemplar self-similar scene, which is provided as a set of posed 2D images. We desire that our method produces a plausible 3D scene structure that matches the exemplar scene on a patch basis, and realistic 2D image renderings that are consistent across the space of all views of that scene.

Given this goal, we choose to directly generate a 3D representation, so that we are guaranteed view consistency when rendering 2D images from it. The first question is, *what 3D representation should we use?* One choice would be coordinate-based MLPs, which have been shown to be compact scene representations able to generate very high quality novel views [34]. However, such a representation is ill-suited for a generative setup, as the costly evaluation makes rendering volumetric and image patches in the training loop infeasible, and the global and distributed nature of the MLP representation makes GAN training challenging in our patch-based setting. Another option is to operate directly on discretized RGBA volumes but, as demonstrated in Karnewar et al. [64], i.e., chap. 4, such an approach results in limited quality (blurry) results. Instead, we build the 3INGAN approach using the previously described grid-based ReLU Field representation. This representation is revisited very briefly in sub-sec. 5.3.1.

A naïve extension of SinGAN [151] to 3D fails to produce good results for multiple reasons:

- only using a 3D discriminator doesn't have the notion of free-space in the volume and also tries to replicate the inside regions of the occupied space that may contain random values. When such a model is rendered, the results have shape distortions and chromatic noise.
- 2. only using 2D discriminators on the rendered images is challenging. Specifically, when trained on too small patches the discriminator becomes too weak to inform the generator of the underlying 3D shape, and when trained on too large patches, the generator simply memorizes the initial reconstruction.



Figure 5.2: 3INGAN setup. Overview of our approach with two parts: an initialization of a reference 3D feature grid (top) and a stage-wise learning of a generative model (bottom). Input to the system is a set of 2D images seen on the top left. From these, optimization using differentiable rendering for known views produces the reference feature grid, which is the input to the next step. The rows below ("Level") denote levels of training the generator, a 2D discriminator, and a 3D discriminator. The 3D discriminator (right) gets random 3D patches from the reference or generated 3D grid, while the 2D discriminator (right) gets random 2D patches from reference or from generated renderings.

Our solution involves two main ingredients: the use of ReLU-Fields representation [64] instead of the standard RGBA volumes for inherently inducing the notion of free space in the 3D-grid; and a mixture of 2D and 3D discriminators, along with multi-scale training, that robustly produce consistent and high quality reconstructions.

Method overview. Input to our method is a set of *n* 2D images $\mathcal{I} := \{I_1, \ldots, I_n\}$ taken from a single real world or synthetic self-similar scene. Fig. 5.2 presents a graphical overview and alg. 4 presents a pseudo-code version. Our method consists of two main steps. First, we convert the 2D image set into a 3D feature grid *V* (para. Optimization of subsec. 5.3.1). Then, we train a generative model *G* of 3D scenes from this 3D feature grid and its 2D rendered images (sub-sec. 5.3.2). This generative model $G(\mathbf{z})$ then converts spatial random latent grids (\mathbf{z}) into 3D feature grids containing remixes of the exemplar scene, which can be consistently rendered

Algorithm 4 Our 3INGAN training. Function sample(...) samples all distributions provided as arguments; up(a,b) ADAM-updates the parameters a by the gradient of expression b with respect to a.

1:	repeat	▷ 3D representation building
2:	$\{I,C\}:=\mathtt{sample}(\mathcal{I},\mathcal{C})$	
3:	$V := up(V, \ \zeta(V, C) - I\ _2^2)$	
4:	until converged.	
5:	repeat	▷ Generator training
6:	$\{\mathbf{z},C\} := \mathtt{sample}(\mathcal{N},\mathcal{D})$	
7:	$P := \mathcal{P}_{\mathrm{2D}}(\zeta(G^{m{ heta}}(\mathbf{z}),C))$	
8:	$\phi:= { t up}(\phi, D^{\phi}_{2{ extsf{D}}}(P={ t fake}))$	▷ 2D fake disc. update
9:	$C := \mathtt{sample}(\mathcal{D})$	
10:	$P := \mathcal{P}_{2D}(\zeta(V, C))$	
11:	$\phi := \mathrm{up}(\phi, D_{2\mathrm{D}}^{\phi}(P = \texttt{real}))$	⊳ 2D real disc. update
12:	$\mathbf{z} := \mathtt{sample}(\mathcal{N})$	
13:	$P := \mathcal{P}_{3D}(G^{\hat{\theta}}(\mathbf{z}))$	
14:	$\psi := \mathrm{up}(\psi, D_{\mathrm{3D}}^{\psi'}(P = \mathtt{fake}))$	⊳ 3D fake disc. update
15:	$P := \mathcal{P}_{3\mathrm{D}}(V)$	
16:	$\psi := \operatorname{up}(\psi, D_{3\mathrm{D}}^{\psi}(P = \texttt{real}))$	⊳ 3D real disc. update
17:	$\{\mathbf{z},C\}:=\mathtt{sample}(\mathcal{N},\mathcal{D})$	⊳ Generator update
18:	$ heta:= \mathrm{up}(heta, D^{\phi}_{2\mathrm{D}}(\mathcal{P}_{2\mathrm{D}}(\zeta(G^{ heta}(\mathbf{z}),C)) = \mathtt{rea})$	$(1)) \qquad \qquad \triangleright \text{ from 2D disc.}$
19:	$\theta := up(\theta, D_{3D}^{\psi}(\mathcal{P}_{3D}(G^{\theta}(\mathbf{z})) = real))$	⊳ from 3D disc.
20:	until converged.	

from arbitrary views.

5.3.1 Representation

Foreground ReLU Field [64]. Foreground is bound by a user-provided AABB covered by a volumetric grid, V, of fixed resolution $n_{Vx} \times n_{Vy} \times n_{Vz}$ to contain feature values in the [-1,1] range. These values correspond to the raw-features that are stored on the voxel-grid. In order to obtain continuous density field, the trilinearly interpolated values of these raw features are passed through a single channel-wise ReLU to convert them to [0,1] range which can be physically interpreted as the density values. We do not model view-dependent appearance, i.e., we approximate the scene with Lambertian materials.

Background. The background is assumed to be constant black for synthetic scenes. For real scenes, we model the background of the scene using an implicit neu-
ral network *B*, similar to NeRF++ [106], but without using the inverted-sphere parametrization of the scene. Our goal is not to model the entire scene perfectly, but rather to provide appropriate inductive bias to the reconstruction pipeline to do the foreground-background separation correctly. This allows us to reconstruct real-world scenes without the requirement of additional segmentation masks.

Optimization. Let the camera pose (extrinsic translation and rotation as well as intrinsics) for each input image be $C := \{C_1, ..., C_n\}$ and assume they are known, e.g., by using structure-from-motion (we use ColMap [75]). Further, we denote the rendering operation to convert the feature grid *V* and the camera pose C into an image as $\zeta(V, C)$. Specifically, we use emission-absorption raymarching [72, 97]. See supplementary for more details for the rendering. We can then directly optimize the photometric loss over the feature grid, given the pose and the 2D images as,

$$\arg\min_{V} \sum_{i=1}^{n} \|I_i - \zeta(V, C_i)\|_2^2.$$
(5.1)

We minimize this loss with batched optimization over 2048 random rays out of all the rays for which we know the 2D input image pixel value. Input images are of size 512×512 . Further, instead of directly optimizing for the full-resolution volume V, training proceeds progressively in a coarse-to-fine manner. Initially, the feature grid is optimized at a resolution where each dimension is smaller by factor 16. After seeing 20k batches of input rays, the feature grid resolution is multiplied by two and the feature grid tri-linearly upsampled.

5.3.2 Generation

Training the generative model makes use of the 3D feature grid V trained in the previous sub-section 5.3.1, which we denote as the *reference* grid herein. We look into the generator details first, before explaining the losses used to train it: 2D and 3D discriminators, and a 2D and 3D reconstruction loss.

Generator. Recall, that the model *G* maps random latent codes \mathbf{z} to a 3D feature grid $G(\mathbf{z})$ at the coarsest stage. While adds fine residual details to previous stage's outputs at the rest of the stages similar to SinGAN [151]. The generator is a 3D CNN

(Convolutional Neural Network) that stagewise decodes a spatial grid of noise vectors \mathbf{z} of size n_z (we use *seedDimension* = 4) into the grid of the desired resolution.

Training. We train the architecture progressively: the generator first produces grids of reduced resolution. Only once this has converged, layers are added and the model is trained to produce the higher resolution. Note that we freeze the previously trained layers in order to avoid the GAN training from diverging. We employ an additional reconstruction loss that enforces one single fixed seed \mathbf{z}^* to map to the reference grid. We supervise this fixed seed loss via an MSE over the 3D grids and with 2D rendered patches.

2D discriminator. A 2D loss discriminates 2D patches of renderings of the generated feature grid from 2D patches rendered from the reference grid V. To render the 3D grids we need to model another distribution of poses, denoted by \mathcal{D} , that uniformly samples camera locations to point at the center of the hemisphere and where focal length is varied stagewise linearly, where the value at the final stage corresponds to the actual camera intrinsics. Further, let $\mathcal{P}_{2D}()$ be an operator to extract a random patch from a 2D image, with discriminators $p_{\rm F}^{\rm 2D}$ and $p_{\rm R}^{\rm 2D}$. They are defined as:

$$p_{\rm F}^{\rm 2D} = \mathcal{P}_{\rm 2D}(\zeta(G(\mathbf{z}), \mathcal{D})) \text{ and } p_{\rm R}^{\rm 2D} = \mathcal{P}_{\rm 2D}(\zeta(V, \mathcal{D})).$$
(5.2)

Note, that we did *not* define $p_{\rm R}^{2\rm D} = \mathcal{P}_{2\rm D}(\mathcal{I})$, as this would limit ourselves to use real samples only from the limited set of known 2D image patches. "Trusting" our reference 3D feature grid has been extracted properly, we can instead sample it from arbitrary views and get a much richer set.

3D discriminator. The 3D discriminator compares 3D patches from the generated feature grid to 3D patches of the reference feature grid. Let $\mathcal{P}_{3D}(V)$ be an operator to extract a random patch from a 3D feature grid V. The distributions to discriminate are,

$$p_{\rm F}^{\rm 3D} = \mathcal{P}_{\rm 3D}(G(\mathbf{z}))$$
 and $p_{\rm R}^{\rm 3D} = \mathcal{P}_{\rm 3D}(V).$ (5.3)

Finally, we use Wasserstein GAN [200] to discriminate the corresponding distributions as well as the reconstruction losses in both 2D and 3D,

$$\mathcal{L} = \gamma_{2D} \cdot \text{wgan}(p_{R}^{3D}, p_{F}^{3D}) + \gamma_{3D} \cdot \text{wgan}(p_{R}^{2D}, p_{F}^{2D}) + \rho_{2D} \cdot \mathbb{E}_{\mathsf{C} \in \mathcal{D}}[\|\zeta(G(\mathbf{z}^{\star}), \mathsf{C}) - \zeta(V, \mathsf{C})\|_{2}^{2}] + \rho_{3D} \cdot \|G(\mathbf{z}^{\star}) - V\|_{2}^{2},$$
(5.4)

weighted by two pairs of two factors γ_{2D} , γ_{3D} and ρ_{2D} , ρ_{3D} . In practice, we use $\gamma_{2D} = \gamma_{3D} = 1.0$ and as well as $\rho_{2D} = \rho_{3D} = 10$.

5.4 Evaluation

We perform quantitative and qualitative evaluations of our approach on a number of different scenes. We compare our method to prior works on 3D generative modelling as baselines, and evaluate design choices via an ablation study.

Comparisons and ablations. We compare to four methods (tab. 5.1). Besides our full method 3INGAN (Ours), we study two prior approaches and two ablations.

As there was no existing method, at the time of the publication, for 3D single scene remixing, we instead compared to two recent methods that were designed to learn a 3D generative model for *classes* of objects, trained on a dataset of images/renderings where each instance (object) is seen from multiple different views, PiGAN [116] and Graf [117]. In these baselines, we test how well such methods work when given instead, many rendered views of a single scene. In both cases, we use the code provided by the authors and their recommended parameter settings. We mark whether the original approach was designed for a single scene or not in the last column in tab. 5.1.

In the first ablation, our 2D only ablation, we evaluate the importance of the 3D discriminator and 3D reconstruction seed losses by running a version of our method with those losses removed, i.e. ($\rho_{3D} = \gamma_{3D} = 0$ in eqn. 5.4). We refer to this method as OursPlatoGAN, as its GAN loss setting, which is only on the rendered 2D images, is similar to the setup used in Henzler et al. [97], while being applied on only a single scene.



Figure 5.3: Datasets. Example renderings of the scenes from our synthetic and real world datasets (Blocks, Chalk).

 Table 5.1: Comparisons and ablations. We enumerate the different methods based on how they make use of 2D versus 3D information, and if they operate on a single scene or multiple scenes.

	Di	sc.	Rec	con.	Single	
	2D	3D	2D	3D	scene	
PiGAN [116]	\checkmark	×	×	×	×	
Graf [117]	\checkmark	×	×	×	×	
OursPlatoGAN	\checkmark	×	\checkmark	×	\checkmark	
OursSinGAN3D	×	\checkmark	×	\checkmark	\checkmark	
Ours	\checkmark	\checkmark	\checkmark	\checkmark	\checkmark	

In the second ablation, our 3D only ablation, we evaluate the importance of the 2D discriminator and 2D reconstruction seed loss, which we refer to as OursSin-GAN3D. This approach is a naïve extension of SinGAN [151] to 3D. In other words, it is our approach without any differentiable rendering, i.e., with the 2D discriminator and 2D reconstruction seed losses removed ($\gamma_{2D} = \rho_{2D} = 0$). Please visit the webpage https://geometry.cs.ucl.ac.uk/group_website/projects/2022/prj-animesh_research_blog/project_pages/leaderboard_pages/thr3inGan/results_index.html for further ablations of the reconstruction seed losses.

Scenes. We consider a mix of synthetic and real scenes, with various levels of stochasticity (a requirement for patch-based remixing of scenes). These scenes include a synthetic scene rendered from Blender showing fishes with the same orientation (Fish) as well as with random orientations (FishRot), a scene composed of 3D balloons (Balloons), inspired by [151]. We also use four semi-synthetic scenes that were real scenes reconstructed from images using photogrammetry and then cleaned by an artist and sold on SketchFab: a pile of dirt (DirtPile), a log pile (Logs), and a bush (Plants). We also make a fully synthetic (Forest) scene which has a ground plane so as to resemble most real-world settings. Finally, we include two

5.4. Evaluation

real scenes with background for which we have no ground truth 3D available, which both show a random arrangement of geometric toys (Blocks) or pieces of colored chalks (Chalk). Note that in all the cases, regardless of the source, our method only accesses 2D renderings/images of the scene, not the 3D scene.

Evaluation metric. We evaluate our method along two axes – *visual quality*, and *scene diversity*. With traditional 2D GANs, these are most commonly evaluated using FID. However, this metric is typically used over two datasets of images, whereas our situation is slightly different; we have only one ground truth scene, and a diverse distribution of generated scenes. We extend SIFID [151] to 3D, and also explicitly separate quality and diversity to separately compare along each axis.

Visual quality is measured as the expectation of SIFID scores between the distribution of exemplar 2D *images* and the distribution of rendered generated 2D images for a fixed camera over multiple seeds. We compute this expectation by taking a mean over a number of camera-views. This is similar in sprit to SIFID, except we compute it over images rendered from different views of the 3D scenes. Lower distance reflects better quality.

Unfortunately, in the single scene case, we *cannot* meaningfully compute FID scores between the exemplar patch distribution and the distribution of all generated patches across seeds as it would make diversity appear as a distribution error. This is different from a typical GAN setup where we are given real images as samplings of desirable distribution of both quality and diversity. Instead, we measure *scene diversity* as the variance of a fixed patch from a fixed view over random seeds, which is a technique used to study texture synthesis diversity [201]. Larger diversity is better.

5.4.1 Qualitative Evaluation

In fig. 5.5 and fig. 5.6, we present qualitative results of 3INGAN, prior methods, and ablations. We can see that, by construction, our method produces variations of 3D scenes that are view consistent.

Visual quality. While some artifacts remain, we observe that this task is challenging for all existing approaches, and when compared to prior work and simple baselines,



Figure 5.4: Single 3D scene FID. We extend FID scores to our single scene use case. The distribution of feature responses is computed for different camera views (rows) and generations (columns). The reference (left) leads to a certain distribution of features. Rather than matching the reference distribution across all views and all seeds (red lines), we compare it to the distribution of a single fixed seed across all views (green lines) to measure *visual quality*. Then, we compare the variance of the distribution of features across all seeds under a fixed view as a measure of *scene diversity*.



Figure 5.5: Qualitative comparison. Comparison of visual quality for different methods (columns) for different scenes (rows).

5.4. Evaluation

Table 5.2: Quality versus diversity. A good generative model should have a good mix of quality and diversity – excellent quality with no diversity or vice versa are both undesirable. Visual Quality and Scene Diversity for different methods (columns) and different data sets (rows). To simplify comparison, we normalize the numbers so that ours is always 1. The best for each metric on each dataset is **bolded** and second best is <u>underlined</u>. Please refer to the supplementary for unscaled numbers.

	PiGAN	PiGAN [116]		Graf [117]		toGAN	OursSin	GAN3D	Ours	
	Qual. \downarrow	Div. ↑	Qual. \downarrow	Div. \uparrow	Qual. \downarrow	Div. \uparrow	Qual. \downarrow	Div. ↑	Qual. \downarrow	Div. ↑
Fish	15.74	0.261	264.27	0.359	465.47	2.282	<u>8.98</u>	0.332	1.00	1.000
FishR	2.47	0.440	3.07	<u>1.018</u>	46.02	2.499	<u>2.70</u>	0.170	1.00	1.000
Balloons	1.44	0.024	3.25	0.477	1.79	0.482	9.57	0.083	1.00	1.000
Dirt	<u>0.96</u>	0.264	1.69	<u>0.440</u>	0.66	0.131	6.06	0.164	1.00	1.000
Forest	1.33	0.473	1.69	0.801	0.70	<u>0.883</u>	2.14	0.439	1.00	1.000
Plants	<u>5.99</u>	0.261	6.56	<u>0.648</u>	12.81	0.144	9.15	0.326	1.00	1.000
Blocks	<u>0.91</u>	0.224	0.73	0.342	1.83	0.334	3.32	0.329	1.00	1.000
Chalk	0.02	0.061	0.01	0.320	0.54	0.088	0.85	0.035	1.00	1.000



Figure 5.6: Diversity across different generative samples. Diversity under changing seeds (columns) of different methods (rows) for different scenes (left and right blocks). See also Figure 5.3.

our method achieves higher visual quality. As expected, scenes with higher stochasticity result in better visual quality and diversity. For example, the balloons are mostly intact but shifted to different locations, and the dirt pile consists largely of reasonable structures.

5.4. Evaluation

Scene diversity. Compared to the competing baselines, PiGAN and Graf, we observe that 3INGAN obtains significantly more scene diversity. This is not surprising, as prior methods have been designed to work on multi-scene image collections, and hence experience mode collapse when given views of only a single scene. This motivates the development of a patch-based generative model for 3D scenes. We see that 3INGAN learns to keep the identity of objects (e.g., balloons, fish, chalk) and create plausible 3D variations (e.g., balloons floating in air, or blocks meaningfully stacked). Please refer to the supplementary for more results.



Figure 5.7: Scene retargetting. Retargeting the Plants, the Logs, and Fish scenes to novel aspect ratios. Since ours is CNN-based, it is easy to retarget scenes to different sizes.

Scene retargeting. In fig. 5.7 we show results of changing the aspect ratio of generated scenes. As our generator is fully convolutional, this can be done simply by changing the shape of the input noise. We can see that the scene structure remains plausible, with the semi-stochastic content repeated to fill in the space.

5.4.2 Quantitative Evaluation

In tab. 5.2, we quantitatively compare the different variations using our proposed metrics. Mirroring the qualitative observations, we can see that 3INGAN performs better than prior work and ablations in terms of visual quality. The numbers show the importance of having both the 2D and 3D discriminators as they help to improve the object appearance and geometric structure, respectively, of the generations. Our main advantage is the noticeable boost in diversity of the generated results, as also measured in the $2-5 \times$ boost in scene diversity score over the other methods. Note that as indicated by the relative *Visual Quality* scores, our results are not yet close to being photorealistic (indistinguishable from the reference distribution) but nonetheless we achieve a healthy gain in over quality/diversity over competing alternatives.

5.5 Limitations

Our approach has a number of limitations. For one, we note that the proposed method of generating a distribution (i.e., a generative model) from a single scene only makes sense in case where scenes contain stochastic structures that can be shuffled around. We do not expect our method to work on highly structured content, e.g., people. Furthermore, we observe that results can still exhibit artifacts, such as high frequency noise, blobby output, and floaters. We believe that this is because the scene distribution is being estimated from very limited data (i.e., a single scene), and is analogous to the "splotchyness" commonly seen in 2D GANs when trained in limited data settings or over complex distributions. In this case, it should be possible to improve quality by using data augmentation as recommended in [135], or by improving the training regime, as in [152], however in the latter case it is important to balance diversity to avoid mode collapse. Finally, in our implementation, we assume scenes to be Lambertian, i.e., without view-dependent effects. In the future, view-dependent specular effects could be modelled using SH components.

5.6 Summary

We have presented 3INGAN, a method for training a generative model of remixes of a single 3D scene. Similar to SinGAN, our approach works on "long-tail" data, as it does not require aligned 3D datasets for training. Instead, we require only a set of posed images of a single self-similar scene as input. We believe that this method has a number of downstream applications, such as retargetting or harmonization [151], and furthermore, studies in generative models for 3D scenes may, one day, lead to realistic view-consistent scene generation on par with images. This would be practical not only for 3D content generation, but as a generative prior to be used in many 3D reconstruction and editing tasks.

Chapter 6

HoloDiffusion: Training a 3D Diffusion Model using 2D Images

6.1 Background and Contributions

Around this time, Diffusion Models (chapter 2.3.2) were gaining traction in the research community, due to the ground-breaking results produced by works such as Stable-Diffusion [12]. Stable-Diffusion built upon the works of iDDPM [202], guided-diffusion [81], and latent-diffusion [203]. These works pushed forward the sota in 2D text-to-image synthesis, which motivated us to also consider Diffusion Models in our research on 3D synthesis. So, as a tryout, we decided to replace the GAN part in 3inGAN (chapter 5) with a Diffusion Model. This lead to a mini-project titled "3inFusion", the idea of which is described in figure 6.1. Once trained, the 3inFusion model can then be used to generate semantically meaningful "remixes" of the original 3D scene (see figure 6.2). Although this mini-project shares the same problem motivation as 3inGAN, we found the diffusion model to be much easier to work with. Using diffusion as the generative backbone allowed us to use a much simpler architecture here, compared to the multi-scale setup required for 3inGAN, and the models trained much smoothly requiring little-to-no hyperparameter tuning.

Given these observed benefits in 3inFusion, moving on from the 3inGAN project, we decided to pursue diffusion models in the rest of the projects instead of GANs. I then started a Research Scientist internship at Meta under the mentorship



Figure 6.1: The 3inFusion pipeline takes as input random 3D crops of the fitted ReLU-Field grid. This input is then combined with time-conditional Gaussian noise, which is then denoised by the 3D-Unet. During inference, we use this 3D crop denoiser to denoise a noise grid of the original grid-size iteratively to generate semantically meaningful variations of the original scene.

of David Novotny. The overall idea of this research project here was to train a largerscale diffusion model on real-captured 3D data, since the team had just released the now popular Co3D dataset [76]. As obvious, the straightforward approach was to do this in two stages, where we first fit a chosen 3D scene representation for all the G.T. multi-view posed imagery. And, then train a 3D diffusion denoiser model on this corpus of fitted 3D scenes. As it turns out, many parallel works such as DiffRF [185] and RODIN [184] proposed this exact approach. However, it is still quite puzzling that both of these methods apply 2D render-losses for training the diffusion models in spite of carrying out the computationally expensive fitting stage. So, I hypothesized that perhaps this fitting stage is redundant, and proposed a harder version of the problem in that: "can we train a 3D diffusion model using only multi-view 2D images?". This question motivated and drove this project titled HoloDiffusion, in which we do propose such a pipeline that can train 3D diffusion models while only requiring 2D multi-view image based datasets. Such a method is a perfect fit for the Co3D dataset, because the dataset contains of 360° spin-around videos of commonly occurring 3D objects.

Since the Co3D dataset was newly released, and it used a newly developed framework titled Implicitron (https://github.com/facebookresearch/pytorch3d/tree/main/pytorch3d/implicitron), there wasn't much docu-



Figure 6.2: Given a ReLU-field grid of a 3D scene, 3inFusion can generate semantically meaningful variations of it, much better than 3inGAN, while training in a simple and stable manner.

mentation available regarding them. Thus, David helped with the implementation of this idea in Implicitron using the Co3D dataset. Although the research motivation and the core of the idea were proposed by me, David made the key addition of the bootstrapped denoising trick which allowed to obtain correct 3D samples from the trained models. The details regarding this can be found later in the chapter. Because of this, David and I share equal contribution on this publication. Niloy and Andrea provided crucial insights in our research discussions and also contributed to drafting and disseminating the paper.

6.2 Introduction

Diffusion models have rapidly emerged as formidable generative models for images, replacing others (e.g., VAEs, GANs) for a range of applications, including image colorization [204], image editing [205], and image synthesis [206, 81]. These models explicitly optimize the likelihood of the training samples, can be trained on millions if not billions of images, and have been shown to capture the underlying model distribution better [81] than previous alternatives.

6.2. Introduction



Figure 6.3: We present HoloDiffusion as the first **3D-aware generative diffusion model** that produces 3D-consistent images and is trained with only posed image supervision. Here we show a few different samples generated from models trained on different classes of the CO3D dataset [76].

A natural next step is to bring diffusion models to 3D data. Compared to 2D images, 3D models facilitate direct manipulation of the generated content, result in perfect view consistency across different cameras, and allow object placement using direct handles. However, learning 3D diffusion models is hindered by the lack of a sufficient volume of 3D data for training. A further question is the choice of representation for the 3D data itself (e.g., , voxels, point clouds, meshes, occupancy grids, etc.). Researchers have proposed 3D-aware diffusion models for point clouds [174], volumetric shape data using wavelet features [207] and novel view synthesis [179]. They have also proposed to distill a pretrained 2D diffusion model to generate neural radiance fields of 3D objects [26, 178]. However, a diffusion-based 3D generator model trained using only 2D image for supervision is not available yet.

In this chapter, we contribute HoloDiffusion, the first unconditional 3D diffusion model that can be trained with *only* real posed 2D images. By posed, we mean different views of the same object with known cameras, for example, obtained by means of structure from motion [75].

We make two main technical contributions: (i) We propose a new 3D model that uses a hybrid explicit-implicit feature grid. The grid can be rendered to produce images from any desired viewpoint and, since the features are defined in 3D space, the rendered images are consistent across different viewpoints. Compared to utilizing an explicit density grid, the feature representation allows for a lower resolution grid. The latter leads to an easier estimation of the probability density due to



Figure 6.4: Method overview. Our HoloDiffusion takes as input video frames for categoryspecific videos $\{s^i\}$ and trains a diffusion-based generative model \mathcal{D}_{θ} . The model is trained with only posed image supervision $\{(I_j^i, P_j^i)\}$, without access to 3D ground-truth. Once trained, the model can generate view-consistent results from novel camera locations.

a smaller number of variables. Furthermore, the resolution of the grid can be decoupled from the resolution of the rendered images. (ii) We design a new diffusion method that can learn a distribution over such 3D feature grids while only using 2D images for supervision. Specifically, we first generate intermediate 3D-aware features conditioned only on the input posed images. Then, following the standard diffusion model learning, we add noise to this intermediate representation and train a denoising 3D UNet to remove the noise. We apply the denoising loss as photometric error between the rendered images and the Ground-Truth training images. The key advantage of this approach is that it enables training of the 3D diffusion model from 2D images, which are abundant, sidestepping the difficult problem of procuring a huge dataset of 3D models for training.

We train and evaluate our method on the Co3Dv2 [76] dataset where HoloDiffusion outperforms existing alternatives both qualitatively and quantitatively.

6.3 HoloDiffusion method

6.3.1 Learning 3D Categories by Watching Videos

Training data. The input to our learning procedure is a dataset of $N \in \mathbb{N}$ video sequences $\{s^i\}_{i=1}^N$, each depicting an instance of the same object category (e.g., , car, carrot, teddy bear). Each video $s^i = (I_j^i, P_j^i)_{j=1}^{N_{\text{frames}}}$ comprises N_{frames} pairs (I_j^i, P_j^i) ,

each consisting of an RGB image $I_j^i \in \mathbb{R}^{3 \times H \times W}$ and its corresponding camera pose $P_j^i \in \mathbb{R}^{4 \times 4}$, represented as a 4 × 4 camera matrix. Our goal is to train a generative model p(V) where V is a representation of the shape and appearance of a 3D object; furthermore, we aim to learn this distribution using only the 2D training videos $\{s^i\}_{i=1}^N$.

3D feature grids. As 3D representation *V* we pick *3D feature voxel grids* $V \in \mathbb{R}^{d^V \times S \times S \times S}$ of size $S \in \mathbb{N}$ containing d^V -dimensional latent feature vectors. Given the voxel grid *V* representing the object from a certain video *s*, we can reconstruct any frame $(I_j, P_j) \in s$ of the video as $I_j = r_{\zeta}(V, P_j)$ by the means of the *rendering function* $r_{\zeta}(V, P_j) : \mathbb{R}^{d^V \times S \times S \times S} \times \mathbb{R}^{4 \times 4} \mapsto \mathbb{R}^{3 \times H \times W}$, where ζ are the function parameters. Next, we discuss how to build a diffusion model for the distribution p(V) of feature grids. One might attempt to directly apply the methodology of base diffusion models, setting x = V, but this does not work because we have no access to ground-truth feature grids *V* for training; instead, these 3D models must be inferred from the available 2D videos while training. We solve this problem in the next section.

6.3.2 Bootstrapped Latent Diffusion Model

In this section, we show how to learn the distribution p(V) of feature grids from the training videos *s* alone. In what follows, we use the symbol \mathcal{V} as a shorthand for p(V). The training videos provide RGB images *I* and their corresponding camera poses *P*, but no sample feature grids *V* from the target distribution \mathcal{V} . As a consequence, we also have no access to the noised samples $V_t \sim \mathcal{N}(\sqrt{\bar{\alpha}_t}V_0, (1-\bar{\alpha}_t)I)$ required to evaluate the denoising objective and thus learn a diffusion model. To solve this issue, we introduce the BLDM (*Bootstrapped Latent Diffusion Model*). BLDM can learn the denoiser-cum-generator \mathcal{D}_{θ} given samples $\bar{V} \sim \bar{\mathcal{V}}$ from an *auxiliary distribution* $\bar{\mathcal{V}}$, which is closely related but not identical to the target distribution \mathcal{V} .

The auxiliary samples \bar{V} . As shown in 6.4, our idea is to obtain the auxiliary samples \bar{V} as a (learnable) function of the corresponding training videos *s*. To this end, we use a design strongly inspired by Warp-Conditioned-Embedding (WCE) [208], which demonstrated compelling performance for learning 3D object categories. Specifically, given a training video *s* containing frames I_j , we generate a grid $\bar{V} \in \mathbb{R}^{d^V \times S \times S \times S}$ of

auxiliary features $\bar{V}_{:mno} \in [-1, 1]^{d_V}$ by projecting the 3D coordinate \mathbf{x}_{mno}^V of the each grid element (m, n, o) to every video frame I_j , sampling corresponding 2D image features, and aggregating those into a single d_V -dimensional descriptor per grid element. The 2D image features are extracted by a trainable encoder (we use the ResNet-32 encoder [43]) *E*.

Auxiliary denoising diffusion objective. The standard denoising diffusion loss is unavailable in our case because the data samples V are unavailable. Instead, we leverage the " x_0 -formulation" of diffusion to employ an alternative diffusion objective which does not require knowledge of V. Specifically, we replace with a *photometric loss*

$$\mathcal{L}_{\text{photo}} := \| r_{\zeta}(\mathcal{D}_{\theta}(\bar{V}_t, t), P_j) - I_j \|^2, \tag{6.1}$$

which compares the rendering $r_{\zeta}(\mathcal{D}_{\theta}(\bar{V}_t,t),P_j)$ of the denoising $\mathcal{D}_{\theta}(\bar{V}_t,t)$ of the noised auxiliary grid \bar{V}_t to the (known) image I_j with pose P_j . The diffusion loss can be computed because the image I_j and camera parameters P_j are known and \bar{V}_t is derived from the auxiliary sample \bar{V} , whose computation is given in the previous section.

Train/test denoising discrepancy. Our denoiser \mathcal{D}_{θ} takes as input a sample \bar{V}_t from the noised *auxiliary* distribution \bar{V}_t instead of the noised *target* distribution \mathcal{V}_t . While this allows to learn the denoising model by minimizing the diffusion loss, it prevents us from drawing samples from the model at test time. This is because, during training, \mathcal{D}_{θ} learns to denoise the auxiliary samples $\bar{V} \in \bar{\mathcal{V}}$ (obtained through fusing image features into a voxel-grid), but at test time we need instead to draw target samples $V \in \mathcal{V}$ as specified by the sampling equation per sampling step. We address this problem by using a bootstrapping technique that we describe next.

Two-pass diffusion bootstrapping. In order to remove the discrepancy between the training and testing sample distributions for the denoiser \mathcal{D}_{θ} , we first use the latter to obtain 'clean' voxel grids from the training videos during an initial denoising phase, and then apply a diffusion process to those, finetuning \mathcal{D}_{θ} as a result. Our bootstrapping procedure rests on the assumption that once $\mathcal{L}_{\text{photo}}$ is minimized, the denoisings $\mathcal{D}_{\theta}(\bar{V}_t, t)$ of the auxiliary grids $\bar{V} \sim \bar{V}$ follow the clean data distribution



Figure 6.5: View consistency. Evaluation of the consistency of the shape renders under camera motion. While our results (top) remain consistent, pi-GAN [116]'s results (bottom) suffer from significant appearance variations across view changes.

 \mathcal{V} , i.e., $\mathcal{D}_{\theta^{\star}}(\bar{V}_t,t) \sim \mathcal{V}$ for the optimal denoiser parameters θ^{\star} that minimize \mathcal{L}_{photo} . Simply put, the denoiser \mathcal{D}_{θ} learns to denoise *both* the diffusion noise and the noise resulting from imperfect reconstructions. Note that our assumption $\mathcal{D}_{\theta^{\star}}(\bar{V}_t,t) \sim \mathcal{V}$ is reasonable since recent single-scene neural rendering methods [34, 63, 209] have demonstrated successful recovery of high-quality 3D shapes solely by optimizing the photometric loss via differentiable rendering. Given that $\mathcal{D}_{\theta^{\star}}$ is now capable of generating clean data samples, we can expose it to the noised version of the clean samples *V* by executing a second denoising pass in a recurrent manner. To this end, we define the *bootstrapped photometric loss* \mathcal{L}'_{photo} :

$$\mathcal{L}_{\text{photo}}' := \| r_{\zeta} (\mathcal{D}_{\theta}(\varepsilon_{t'}(\mathcal{D}_{\theta}(\bar{V}_t, t), t'), P_j) - I_j \|^2$$
(6.2)

with $\varepsilon_{t'}(Z) \sim \mathcal{N}(\sqrt{\bar{\alpha}_{t'}}Z, (1-\bar{\alpha}_{t'})I)$ denoting the diffusion of input grid *Z* at time *t'*. Intuitively, the two pass bootstrapping equation evaluates the photometric error between the ground truth image *I* and the rendering of the doubly-denoised grid $\mathcal{D}_{\theta}(\varepsilon_{t'}(\mathcal{D}_{\theta}(\bar{V}_{t},t),t')).$

6.3.3 Implementation Details

Training details. HoloDiffusion training finds the optimal model parameters θ , ζ by minimizing the sum of the photometric and the bootstrapped photometric losses $\mathcal{L}_{photo} + \mathcal{L}'_{photo}$ using the Adam optimizer with an initial learning rate $5 \cdot 10^{-5}$ (decaying ten-fold whenever the total loss plateaus) until convergence is reached. In each training iteration, we randomly sample 10 source views $\{I_j\}$ from a randomly selected training video s^i to form the grid of auxiliary features \bar{V} . The auxiliary features are noised to form \bar{V}_t and later denoised with $\mathcal{D}_{\theta}(\bar{V}_t)$. Afterwards $\mathcal{D}_{\theta}(\bar{V}_t)$ is noised and denoised again during the two-pass bootstrap procedure. To avoid two rendering passes in each training iteration (one for \mathcal{L}_{photo} and the second for \mathcal{L}'_{photo}), we randomly choose to optimize the \mathcal{L}'_{photo} with 50-50 probability in each iteration as a lazy regularization. The photometric losses compare renders $r_{\zeta}(\cdot, P_j)$ of the denoised voxel grid to 3 target views (different from the source views).

Rendering function r_{ζ} . The differentiable rendering function $r_{\zeta}(V, P_j)$ uses Emission-Absorption (EA) ray marching as follows. First, given the knowledge of the camera parameters P_j , a ray $\mathbf{r_u} \in S^2$ is emitted from each pixel $\mathbf{u} \in \{0, ..., H-1\} \times \{0, ..., W-1\}$ of the rendered image $\hat{I}_j \in \mathbb{R}^{3 \times H \times W}$. We sample N_S 3D points $(\mathbf{p}_i)_{i=1}^{N_S}$ on each ray at regular intervals $\Delta \in \mathbb{R}$. For each point \mathbf{p}_i , we sample the corresponding voxel grid feature $V[\mathbf{p}_i] \in \mathbb{R}^{d^V}$, where $V[\cdot]$ stands for trilinear interpolation. The feature $V[\mathbf{p}_i]$ is then decoded by an MLP as $f_{\zeta}(V[\mathbf{p}_i], \mathbf{r_u}) := (\sigma_i, \mathbf{c}_i)$ with parameters ζ to obtain the density $\sigma_i \in [0, 1]$ and the RGB color $\mathbf{c}_i \in [0, 1]^3$ of each 3D point. The MLP f is designed so that the



Figure 6.6: Comparisons. Samples generated by our HoloDiffusion compared to those by pi-GAN, EG3D, and GET3D.

color **c** depends on the ray direction $\mathbf{r}_{\mathbf{u}}$ while the density $\boldsymbol{\sigma}$ does not, similar to NeRF [34]. Finally, EA ray marching renders the $\mathbf{r}_{\mathbf{u}}$'s pixel color $\mathbf{c}_{\mathbf{r}_{\mathbf{u}}} = \sum_{i=1}^{N_S} w(\mathbf{p}_i) \mathbf{c}_i$ as a weighted combination of the sampled colors. The weights are defined as $w(\mathbf{p}_i) = T_i - T_{i+1}$ where $T_i = e^{-\sum_{i=1}^{i-1} \sigma_i \Delta}$.

6.4 Evaluation

In this section, we evaluate our method. First we perform the *quantitative* evaluation and then follow it by visualizing samples for assessing the *quality* of generations.

Datasets and baselines. For our experiments, we use CO3Dv2 [76], which is currently the largest available dataset of fly-around real-life videos of object categories. The dataset contains videos of different object categories and each video makes a complete circle around the object, showing all sides of it. Furthermore, camera poses and object foreground masks are provided with the dataset (they were obtained by the authors by running off-the-shelf Structure-from-Motion and instance segmentation software, respectively).

We consider the four categories Apple, Hydrant, TeddyBear and Donut for our experiments. For each of the categories we train a single model on the 500 "train" videos (i.e. approx. 500×100 frames in total) with the highest camera cloud quality score, as defined in the CO3Dv2 annotations, in order to ensure clean ground-truth camera pose information. We note that all trainings were done on 2-to-8 V100 32GB GPUs for 2 weeks.

We consider the prior works pi-GAN [116], EG3D [169], and GET3D [172] as baselines for comparison. Pi-GAN generates radiance fields represented by MLPs and is trained using an adversarial objective. Similar to our setting, they only use 2D image supervision for training. EG3D [169] uses the feature triplane, decoded by an MLP as the underlying representation, while needing both the images and the camera poses as input to the training procedure. GET3D [172] is another GAN-based baseline, which also requires the images and camera poses for training. Apart from this, GET3D also requires the fg/bg masks for training; which we supply in form of the masks available in CO3Dv2. Since GET3D applies a Deformable Marching Tetrahedra step in the pipeline, the samples generated by them are in the form of textured meshes.

Table 6.1: Quantitative evaluation. FID and KID on 4 classes of CO3Dv2 comparing our HoloDiffusion with the baselines pi-GAN [116], EG3D [169], GET3D [172], and the non-bootstrapped version of our HoloDiffusion. The column "VP" denotes whether renders of a method are 3D view-consistent or not.

method	VP	Apple		Hydrant		TeddyBear		Donut		Mean	
		$FID\downarrow$	$\text{KID}\downarrow$								
pi-GAN [116]	×	49.3	0.042	92.1	0.080	125.8	0.118	99.4	0.069	91.7	0.077
EG3D [169]	\checkmark	170.5	0.203	229.5	0.253	236.1	0.239	222.3	0.237	214.6	0.233
GET3D [172]	\checkmark	179.1	0.190	303.3	0.380	244.5	0.280	209.9	0.230	234.2	0.270
HoloDiffusion (No bootstrap)	\checkmark	342.9	0.400	277.9	0.305	222.1	0.217	272.1	0.199	278.7	0.280
HoloDiffusion	\checkmark	94.5	0.095	100.5	0.079	109.2	0.106	115.4	0.085	104.9	0.091

Quantitative evaluation. We report Frechet Inception Distance (FID) [210], and Kernel Inception Distance (KID) [211] for assessing the generative quality of our results. As shown in quantitave evaluation table, our HoloDiffusion produces better scores than EG3D and GET3D. Although pi-GAN gets better scores than ours on some categories, we note that the 3D-agnostic training procedure of pi-GAN cannot recover the proper 3D structure of the unaligned shapes of CO3Dv2. Thus,



Figure 6.7: Sampling across time. Rendering of HoloDiffusion's iterative sampling process for a hydrant and a teddy bear. The diffusion time decreases from left (t = T = 1000) to the right (t = 0).

without the 3D-view consistency, the 3D neural fields (MLPs) produced by pi-GAN essentially mimic a 2D image GAN.

Qualitative evaluation. Figure 6.6 depicts random samples generated from all the methods under comparison. HoloDiffusion produces the most appealing, consistent and realistic samples among all. Figure 6.6 further analyzes the viewpoint consistency of pi-GAN compared to ours. It is evident that, although individual views of pi-GAN samples look realistic, their appearance is inconsistent with the change of viewpoint. Please refer to the project webpage at https://holodiffusion.github.io for more examples and videos of the generated samples.

6.5 Limitations

At present, our method requires access to camera information at training time. One possibility is to jointly train a viewpoint estimator to pose the input images, but the challenge may be to train this module from scratch as the input view distribution is unlikely to be uniform [212]. An obvious next challenge would be to test the setup for conditional generation, either based on images (i.e., single view reconstruction task) or using text guidance. Beyond generation, we would also like to support editing the generated representations, both in terms of shape and appearance, and compose them together towards scene generation. Finally, we want to explore multi-class training where diffusion models, unlike their GAN counterparts, are known to excel without suffering from mode collapse.

6.6 Summary

We have presented HoloDiffusion, an unconditional 3D-consistent generative diffusion model that can be trained using only posed-image supervision. At the core of our method is a learnable rendering module that is trained in conjunction with the diffusion denoiser, which operates directly in the feature space. Furthermore, we use a pretrained feature encoder to decouple the cubic volumetric memory complexity from the final image rendering resolution. We demonstrate that the method can be trained on raw posed image sets, even in the few-image setting, striking a good balance between quality and diversity of results.

Our method primarily contributes towards the generative modeling of 3D realcaptured assets. Thus as is the case with 2D generative models, ours is also prone to misuse of generated synthetic media. In the context of synthetically generated images, our method could potentially be used to make fake 3D view-consistent GIFs or videos. Since we only train our models on the virtually harmless Co3D (Common objects in 3D) dataset, our released models could not be directly used to infer potentially malicious samples. As diffusion models can be prone to memorizing the training data in limited data settings [213], our models can also be used to recover the original training samples. Analyzing the severity and the extent to which our models suffer from this, is an interesting future direction for exploration.

Chapter 7

HoloFusion: Towards Photo-realistic 3D Generative Modeling

7.1 Background and Contributions

We successfully proposed a method in HoloDiffusion [70] (chapter 6) that can train a 3D diffusion model using only multi-view 2D images as supervision. It produced decent quality 3D view-consistent results, but one of the issues with the approach was visual quality of the samples. Since we aimed to obtain photo-realistic 3D samples using generative modelling on this research journey, we decided to focus on the sample quality aspect of the proposed HoloDiffusion pipeline. Thus, in short, the main goal of this research project was to find an answer to the question: "can we improve the quality of the HoloDiffusion generated samples to photo-realistic level?". We detail various aspects of finding the answer to this question in this chapter. Our proposed solution HoloFusion applies a 3D-2D hybrid diffusion based approach that can produce almost photo-realistic 3D samples while still only requiring mutli-view 2D images as supervision. In order to continue working on this project, I joined the same team at Meta as a visiting researcher along side my PhD fellowship.

I contributed most of the main ideas, the code and the experimentation of this project. David again helped quite a bit with the compute-clusters to run these experiments and also squashed some key bugs in the code that allowed to obtain the photo-realistic results that we showcase in the project. Niloy and Andrea, as



Figure 7.1: We propose HoloFusion to generate photo-realistic 3D radiance fields by extending the HoloDiffusion method with a jointly trained 2D 'super resolution' network. The independently super-resolved images are fused back into the 3D representation to improve the fidelity of the 3D model via distillation, while preserving the consistency across view changes.

usual provided important insights in the research discussions and contributed to the drafting, presentation and dissemination of the paper.

7.2 Introduction

Diffusion-based image generators can now produce high-quality and diverse samples, but their success has yet to fully translate to 3D generation: existing diffusion methods can either generate low-resolution but 3D consistent outputs, or detailed 2D views of 3D objects but with potential structural defects and lacking view consistency or realism. We present HoloFusion, a method that combines the best of these approaches to produce high-fidelity, plausible, and diverse 3D samples while learning from a collection of multi-view 2D images only. The method first generates coarse 3D samples using a variant of the recently proposed HoloDiffusion generator. Then, it independently renders and upsamples a large number of views of the coarse 3D model, super-resolves them to add detail, and distills those into a single, high-fidelity implicit 3D representation, which also ensures view-consistency of the final renders. The super-resolution network is trained as an integral part of HoloFusion, end-to-end, and the final distillation uses a new sampling scheme to capture the space of super-resolved signals. We compare our method against existing baselines, including

DreamFusion, Get3D, EG3D, and HoloDiffusion, and achieve, to the best of our knowledge, the most realistic results on the challenging CO3Dv2 dataset.

Diffusion models [177, 81, 12] are at the basis of state-of-the-art 2D image generators which can now produce very high-quality and diverse outputs. However, their success has yet to be translated to 3D and there is no generator that can produce 3D assets of a comparable quality.

Recent attempts at extending diffusion to 3D generation have reported mixed success. Some authors have attempted to apply diffusion directly in 3D [70], or still in 2D but using a 3D-aware neural network [179, 214]. This requires solving two problems: first, finding a suitable 3D representation (e.g., triplane features [169], mesh [178], voxels [70]) that scales well with resolution and is amenable to diffusion; and, second, obtaining a large amount of 3D training data, for example using synthetic models [184, 185], or training the model using only 2D images [70], often via differentiable (volume) rendering [97, 34]. However, the quality of results so far is limited, especially when training on real images.

Other authors have proposed to *distill* 3D objects from pre-trained 2D image generators. For instance Score Distillation Sampling (SDS) [26] can sample 3D objects from a high-quality off-the-shelf 2D diffusion model while requiring no (re)training. However, without any 3D guidance, distillation methods often produce implausible results; for example, they suffer from the 'Janus effect', where details of the front of the object are replicated on its back. They also create overly-smooth outputs that average out inconsistencies arising from the fact that the signal obtained from the 2D model is analogous to sampling independent views of the object for examples). Furthermore, distillation methods do not support unconditional sampling, even if the underlying image generator does, as strong language guidance is required to stabilise the 3D reconstruction.

In this work, we propose HoloFusion, a method that combines the best of both approaches. We start from HoloDiffusion [70], a diffusion-based 3D generator. This model can be trained using only a multiview image dataset like [76] Co3Dv2 and produces outputs that are 3D consistent. However, the output resolution is limited

by computation and memory. We augment the base model with a lightweight superresolution network that upscales the initial renders. Crucially, the 2D super-resolution model is integrated and trained jointly with the 3D generator, end-to-end.

The super-resolution network outputs detailed views of the 3D object, and the underlying 3D generator ensures that the coarse structure of these views is indeed consistent (e.g., , avoiding the Janus effect and other structural artifacts). However, the 2D upscaling still progresses independently for different views, which means that fine grained details may still be inconsistent between views. We address this issue by distilling a single, coherent, high quality 3D model of the object from the output of the upsampler. For this, we propose a new distillation technique that efficiently combines several putative super-resolved views of the object into a single, coherent 3D reconstruction.

With this, we are able to train a high-quality 3D generator model purely from real 2D data. This model is capable of generating consistent and detailed 3D objects, which in turn result in view-consistent renderings at a quality not achievable by prior methods. We evaluate HoloFusion on real images (CO3Dv2 dataset [76]) and compare with a variety of competing alternatives (e.g., , HoloDiffusion [70], Get3D [172], EG3D [169], DreamFusion [27]) demonstrating that view-consistent high-quality 3D generation is possible using our simple, effective, easy-to-implement hybrid approach.

7.3 HoloFusion method

We present HoloFusion, a method that can learn a high-quality diffusion-based 3D generator from a collection of multiview 2D images. HoloFusion first obtains an unconditional low-resolution 3D sample using diffusion and then distills a high-resolution 3D radiance field representing a higher-quality version of the generated object. We first discuss the low-resolution 3D generator in section 7.3.1 followed by super-resolution distillation in section 7.3.2.



Figure 7.2: Overview. HoloFusion, which trains the 3D denoiser network D_{θ} , is augmented with the 2D 'super-resolution' diffusion model D_{β} . Both models are trained end-to-end by supervising their outputs with 2D photometric error.

7.3.1 HoloDiffusion revisited

Given a large dataset of 3D models, the framework of DDPM [80] could be used to train a corresponding probability distribution. However, such a dataset is not available, and we must instead learn from 2D images of physical 3D objects. Given a dataset containing several views of a large number of objects, we could use image-based reconstruction (e.g., , using neural rendering) to obtain corresponding 3D models first, and then use those to train a diffusion model. Instead, we adopt, and slightly upgrade, the HoloDiffusion method [70], which learns a 3D diffusion model *directly* from the 2D images.

Training data. HoloDiffusion learns from a collection \mathcal{D} of N image sequences $s_i = (I_j^i, C_j^i)_{j=1}^{N_{\text{frame}}}, i = 1, ..., N$, where frame $I_j^i \in \mathbb{R}^{3 \times H \times W}$ is an RGB image and $C_j^i \in \mathbb{R}^{4 \times 4}$ is the corresponding camera projection matrix, collectively defining the motion of the camera.

3D representation and rendering. The shape and appearance of the object are represented by a voxel grid $V \in \mathbb{R}^{d \times S \times S \times S}$ with resolution *S* containing a *d*-dimensional feature vector per voxel. Given a 3D point $\mathbf{p} \in \mathbb{R}^3$, its opacity $\sigma(\mathbf{p}) \in \mathbb{R}_+$ and color $c(\mathbf{p}) \in \mathbb{R}^3_{[0,1]}$ are obtained from the voxel grid by an MLP $M_\eta(V(\mathbf{p}))$ that takes as input the *d*-dimensional feature vector $V(\mathbf{p})$ extracted form the grid via trilinear

101

interpolation [120]. The usual emission-absorption model [34, 72] is then used to implement a differentiable rendering function R_{η} , mapping the voxel grid *V* and the camera viewpoint *C* into an image $\hat{I} = R_{\eta}(V,C)$, where η are the parameters of the MLP.

Training scheme. HoloDiffusion leverages the DDPM framework (revised in the previous paragraphs) to recover the density p(V) over voxel grids x = V encoding plausible real-life objects. In order to train a DDPM on such 3D data, we would need access to ground-truth 3D models V, which are not available. HoloDiffusion addresses this problem by making three changes to DDPM.

First, it replaces the data denoising loss with a photometric reconstruction loss. Given a pair $(I,C) \in s$ from one of the training sequences *s*, it replaces the denoising model objective with $\mathbb{E}_{t,\varepsilon,C} \left[\|I - R_{\eta}(D_{\theta}(\hat{\varepsilon}_t(V),t),C)\|^2 \right]$ where the goal is not to reconstruct the 'clean' volume *V* (which is unknown), but rather its image *I* (which is known).

Second, also because the 'clean' volume V is not available, we cannot use the traditional sampling equation to generate the noisy volumes V_t to denoise; the only exception is the last sample V_T , which is pure noise. This suggests to adopt a 'double denoising' step. First, pure noise V_T is fed into the denoiser to obtain an (approximate) version of $V_0 = D_{\theta}(V_T, T)$ of the clean volume $V_0 = V$. Then, noise is applied to obtain $V_t = \hat{\varepsilon}_t(V_0) = \sqrt{\overline{\alpha}_T}V_0 + \sqrt{1 - \overline{\alpha}_t}\varepsilon_t$ according to traditional sampling equation, and the latter is fed back into the denoiser as above.

Finally, there is the issue that unconditional generation of the clean volume V_0 from pure noise V_T is difficult, especially in the first iterations of training. On the other hand, the problem of *view-conditioned* generation is considerably easier. Hence, the third idea is to learn a *conditional* generator, using a variable number of input views. Specifically, given a training sequence *s*, the method extracts a random subset of frames $\bar{s} \subset s$ (which could be empty, which corresponds to unconditional generation). Then, a feature volume $\bar{V} = \Phi(\bar{s}) \in \mathbb{R}^{d \times S \times S \times S}$ is obtained from the selected frames. This extracts 2D image features using a pre-trained and frozen 2D image encoder and then pools them in 3D via 'unprojection' [125, 208] into \bar{V} , where $\bar{V} = 0$ if \bar{s} is empty. Finally, these pooled features are used to condition the denoising network $V_0 = D_{\theta}(V_T, \bar{V}, T)$, which, on average, leads to a simpler reconstruction problem.

Putting it all together, the training loss becomes:

$$\mathcal{L}(\boldsymbol{\theta}|I, C, \bar{s}) = \mathbb{E}_{t, \hat{\varepsilon}, V_T} \left[\|\hat{I} - I\|^2 \right],$$
(7.1)

where
$$\hat{I} = R_{\eta}(D_{\theta}(V_t, \bar{V}, t), C),$$
 (7.2)

$$egin{aligned} V_t &= \hat{m{\epsilon}}_t(V_0), \ V_0 &= D_{m{ heta}}(V_T, ar{V}, T), \ ar{V} &= \Phi(ar{s}). \end{aligned}$$

This loss is averaged over training sequences s, subsequences $\bar{s} \subset s$, and views $(I,C) \in s$ therein. Note that this is slightly different than the original HoloDiffusion, where feature volume \bar{V} and reconstructed volumes V_t overlap as arguments of the denoiser; we found that keeping them separated in the formulation leads to more stable training and additionally allows for view-conditioned generation.

7.3.2 HoloFusion

The method of section 7.3.1 learns to generate 3D objects from 2D image supervision only, but the fidelity of the output is limited by the resolution at which the operations are carried out. Increasing resolution is difficult due to the GPU memory impact of the voxel-based representation, so we seek a more efficient way to do so. The idea is to incorporate a 2D super-resolution network (section 7.3.2.1), trained end-to-end, that improves the output from the base model. The super-resolved images are eventually fused back in an improved 3D model, which also has the benefit of further increasing view consistency (section 7.3.2.2).

7.3.2.1 Integrating super-resolution

As shown in figure 7.2, we augment the method of section 7.3.1 with a lightweight refinement post-processor network that takes the 2D image \hat{I} generated by the base model and outputs a higher quality version \hat{I}_{super} of the same. This can be thought

of as a form of super-resolution; however, due to the particular statistics of the input ('low-res') images \hat{I} that HoloDiffusion generates, it is necessary to train this super-resolution network in an end-to-end fashion with HoloDiffusion, integrating the two models.

To make this integration seamless, we formulate super-resolution as another diffusion process that runs 'in parallel' with 3D reconstruction. Hence, the super-resolved image $\hat{I}_{super} = D_{\beta}(I_t, \hat{I}, t)$ is the output of a denoiser network (a lightweight U-Net), which takes as input the noised target image $I_t = \hat{\epsilon}_t(I)$ and is also conditioned on the 'low-res' output $\hat{I} = R_{\eta}(V, C)$ of HoloDiffusion from equation 7.2. This denoiser is trained with the DDPM loss:

$$\mathcal{L}(\boldsymbol{\beta}|I) = \mathbb{E}_{t,\hat{\boldsymbol{\varepsilon}}} \left[\| \boldsymbol{D}_{\boldsymbol{\beta}}(\hat{\boldsymbol{\varepsilon}}_t(I), \hat{I}, t) - I \|^2 \right].$$
(7.3)

Training details The overall model $(D_{\beta} \text{ and } D_{\theta})$ is trained end-to-end by optimising the loss $\mathcal{L}(\theta|I, C, \bar{s}) + \mathcal{L}(\beta|I)$ obtained by summing the respective equations.

As training data, we use a large dataset of images capturing object-centric scenes ([76]). In each training batch, we pick a random training scene *s* and sample 15 different source images $\bar{s}_{src} \subset s$ which are unprojected to generate the feature volume conditioning \bar{V} . Then, \bar{V} is rendered into 4 random target views $\bar{s}_{tgt} \subset (s \setminus \bar{s}_{src})$ which allows to optimize the training image reconstruction loss $\mathcal{L}(\theta|I, C, \bar{s}) + \mathcal{L}(\beta|I)$. The latter uses the Adam optimizer with an initial learning rate of $5 \cdot 10^{-5}$ decaying tenfold whenever the loss plateaus until convergence.

7.3.2.2 Fusing super-resolved views in 3D

The method of section 7.3.2.1 leaves us with high-resolution views \hat{I}_{super} of the generated 3D object. However, we would like to obtain a single, high-quality 3D model, not just individual views of it. In this section, we discuss how to take the super-resolved images and fuse them into such a model, while addressing the issues that these images are not perfectly view-consistent.

The basic idea is simple. We can generate a certain number (e.g., 100) high-resolution images of the object from different viewpoints C and then use a technique,



Figure 7.3: Distillation. HoloFusion distills a single high-resolution voxel grid V_0^H by minimizing a top-k patch-remix loss $\mathcal{L}_{\text{distil}}$ between the grid renders $R_{\eta'}(V_0^H, C)$ and a bank \mathcal{I}_C of K = 5 high-res images output by the 2D diffusion upsampler D_β for each scene camera *C*.

akin to neural rendering, to fuse them back into a single 3D model. However, there is a problem with this idea: The model of section 7.3.2.1 generates high-quality views I_{super} , but these are *view-dependent samples* from the distribution $p(\hat{I}_{super}|\hat{I})$ where $\hat{I} = R_{\eta}(V,C)$ is the 'low-res' output form HoloDiffusion. Because super-resolving details is intrinsically ambiguous, there is no reason why samples I_{super} taken from different viewpoints *C* would be consistent (figure 7.7). Fusing them into a single 3D model would then result in a blurry appearance yet again.

As described in figure 7.3, we address this issue in a principled manner by considering *several* possible super-resolved images $\mathcal{I}_C = \{I_{\text{super}} \sim p(\hat{I}_{\text{super}} | \hat{I})\}$ sampled from each given viewpoint *C*. Then, we optimize a high-resolution voxel grid V_0^H by minimizing the photometric loss:

$$\mathcal{L}_{\text{distil}}(\eta', V_0^H | \mathcal{I}_C) = \mathbb{E}_C \left[\min_{I_{\text{super}} \in \mathcal{I}_C} \| I_{\text{super}} - R_{\eta'}(V_0^H, C) \|^2 \right]$$
(7.4)

where $R_{\eta'}(V_0^H, C)$ is the render of a high-resolution voxel grid $V_0^H \in \mathbb{R}^{d \times S' \times S' \times S'}, S' > S$ using the learnable renderer $R_{\eta'}$ with scene specific parameters η' . Minimizing



Figure 7.4: Generated 3D samples visualized from a moving camera. π -GAN and HoloDiffusion* fail to produce 3D view consistent samples, while DreamFusion suffers from the "Janus" problem (multiple heads).

with respect to I_{super} means that the 3D model must be consistent with at least *one* of the possible super-resolved images, drawn from the distribution of super-resolved samples, for each view *C*.

Patch remix. In practice, this approach requires a very large number of super resolved images \mathcal{I}_C to be effective. We found that we can significantly improve the statistical efficiency by performing the minimization at the level of individual patches. Namely, we produce a stack of only $K = |\mathcal{I}_C| = 5$ super resolved images and perform the minimization in equation 7.4 at the level of small 16×16 patches independently (effectively allowing super-resolved images to 'remix' as needed to fit the generated view $R_{\eta'}(V_0^H, C)$).

Distillation details. $\mathcal{L}_{\text{distil}}$ is optimized independently for each generated scene with Adam (lr=2 · 10⁻⁴) for 25K steps until convergence. While η' is initialized using

the pretrained multi-sequence weights η , V_0^H is initialized by trilinearly upsampling the low-resolution volume V_0 output by HoloDiffusion. Cameras *C* are sampled at uniform azimuths with elevation fixed at object's equator.

7.4 Evaluation

We begin with a description of the experiments conducted in subsection 7.4.1, followed by an analysis and discussion of the results in subsection 7.4.2.

7.4.1 Details

Dataset. We experiment on the challenging large-scale Co3Dv2 [76] dataset which is a popular choice for a real-world 3D reconstruction benchmark. More specifically, 4 categories are selected, Apple, Hydrant, TeddyBear, and Donut, with 500 3Dscenes per category for training. Each 3D scene contains ~ 200 images of the object of interest along with poses of their corresponding cameras.

Baselines. We use two sets of baselines for comparison (table 7.1): (i) general 3D generative modeling baselines and (ii) diffusion distillation based baselines. π -GAN [116], EG3D [169], GET3D [172], and HoloDiffusion [70] are considered as the 3D generative baselines. Along with HoloDiffusion, we also test the super-



Figure 7.5: Fusing views. Our patch-remix (section 7.3.2.2) compared to the SDS and MSE distillation. MSE has "floaters" and viewpoint inconsistencies, SDS oversmooths the texture. Ours is robust and produces superior quality.



Figure 7.6: 3D samples generated by our HoloFusion compared to π -GAN, EG3D, GET3D, HoloDiffusion, HoloDiffusion^{*}, and the text-to-3D Stable-DreamFusion.

resolution integrated model HoloDiffusion^{*}. For the distillation-based baselines, we consider the open-source implementation of DreamFusion [26] titled Stable-DreamFusion [27]. For the latter, scenes are generated by conditioning on prompts comprising names of Co3Dv2 categories extended with color and style modifier phrases leading to \sim 200 prompts / 3D shapes per class. More details regarding the prompt creation are in the supplementary.

Table 7.1: FID (↓) and KID (↓) on 4 classes of Co3Dv2 [76]. We compare with 3D generative modeling baselines (rows 1–5); with an SDS distillation-based Stable-DreamFusion (row 6); and with ablations of our HoloFusion (rows 7–8). The column "VP" denotes whether renders of a method are 3D view-consistent or not.

method	VP	Apple		Hydrant		TeddyBear		Donut		Mean	
		$\overline{\text{FID}}\downarrow$	$\text{KID}\downarrow$	$FID\downarrow$	$\text{KID}\downarrow$	$FID\downarrow$	$\text{KID}\downarrow$	$FID\downarrow$	$\text{KID}\downarrow$	$FID\downarrow$	$\text{KID}\downarrow$
π-GAN [116]	×	49.3	0.042	92.1	0.080	125.8	0.118	99.4	0.069	91.7	0.077
EG3D [169]	\checkmark	170.5	0.203	229.5	0.253	236.1	0.239	222.3	0.237	214.6	0.233
GET3D [172]	\checkmark	179.1	0.190	303.3	0.380	244.5	0.280	209.9	0.230	234.2	0.270
HoloDiffusion [70]	\checkmark	94.5	0.095	100.5	0.079	109.2	0.106	115.4	0.085	122.5	0.102
HoloDiffusion*	×	55.9	0.045	62.6	0.045	116.6	0.101	99.6	0.079	83.7	0.068
Stable-DreamFusion [27]	\checkmark	139.0	0.104	185.2	0.132	183.4	0.125	169.3	0.114	169.2	0.119
HoloFusion (MSE)	×	72.7	0.067	62.2	0.045	87.2	0.076	109.0	0.099	82.8	0.072
HoloFusion (SDS)	\checkmark	123.0	0.105	77.1	0.058	117.8	0.090	142.8	0.087	115.2	0.085
HoloFusion (Ours)	\checkmark	69.2	0.063	66.8	0.047	87.6	0.075	109.7	0.098	83.3	0.071

Metrics. We use FID [210] and KID [211] to compare the quality of our 2D renders, as these are commonly used to assess 2D and 3D generators.

7.4.2 Quantitative and qualitative analysis

Table 7.1 evaluates quantitatively while Figure 7.6 qualitatively. Furthermore, Figure 7.4 compares rendering view-consistency.

HoloFusion (Ours) yields better FID/KID scores than the general 3D generative baselines except for π -GAN on Apple and Donut classes. However, since π -GAN does not guarantee view consistency by design, it essentially acts as a 2D image GAN, and thus does better on the 2D FID/KID metrics, but it generates significantly view-inconsistent renders (see figure 7.4 and the supplementary webpage at https://holodiffusion.github.io/holofusion/).

We observed that the other 3D-GAN baselines, EG3D and GET3D, are prone to collapsing to a single adversarial sample leading to poor FID/KID scores. The latter is probably due to the 3D misalignment of the CO3Dv2 sequences across instances, which makes training harder.

HoloFusion also outperforms the text-to-3D Stable-DreamFusion on both FID/KID. Stable-DreamFusion yields good shapes, but produces synthetic-looking and overly-smooth textures and thus performs poorly when compared to the real-world images of Co3Dv2. As evident from the TeddyBear samples, the method also


Figure 7.7: Heatmaps illustrating the per-pixel color variance of K = 10 hypothesis produced by the upsampler D_{β} . Some samples contain artifacts around the object boundaries which correspond to the high-variance regions in the figure. Our top-K patch-remix increases robustness by allowing the loss to discard such artifacts during distillation.

suffers from the "Janus" issue.

Compared to HoloDiffusion, we improve the FID/KID scores by a significant margin, mainly due to the more photo-realistic renders that include high-frequency details. Although the 2D Diffusion upsampler of HoloDiffusion* produces renders with the highest amount of details yielding scores similar to ours, they are not 3D view-consistent (as apparent from figure 7.4 and as explained in section 7.3.2).

Ablations. In table 7.1 and in figure 7.5 we ablate components of our HoloFusion to verify their contribution.

The first variant, HoloFusion (SDS), replaces the Top-k patch-remixed distillation loss with the score distillation sampling (SDS) gradient as proposed in [26]. As apparent from figure 7.5 and from the lower scores, SDS washes out all the high-frequency details in the textures.

Secondly, HoloFusion (MSE) reduces the number of upsampling hypotheses \mathcal{I}

to the minimum of $|\mathcal{I}| = 1$. Even though this slightly improves the 2D metrics, as can be seen from figure 7.5, the samples lack view-consistency and introduce "floaters". In figure 7.7 we further illustrate the variability of the upsampling hypotheses.

7.5 Limitations

Our method suffers from limitations that can be addressed in future work. First, our method is slow to sample from as the sampling process takes about 30 mins for each generation, because it is still a distillation-based method. An interesting extension would be to train another network to directly distill a set of super-resolved images, without requiring explicit optimization during inference. Second, we do not produce an explicit surface representation (e.g., a mesh), which could be done by integrating a differentiable mesh render in the loop as done in some prior work.

7.6 Summary

We have presented a hybrid diffusion-based method that can generate high-quality 3D neural radiance fields of real-life object categories. Our method starts by producing coarse 3D models whose renders are independently super-resolved, and finally consolidated using a robust distillation process. We evaluated our method on the Co3D v2 dataset and presented 3D-consistent, diverse, and high-quality results superior to all competing baselines.

Chapter 8

GOEmbed: Representation Agnostic 3D Feature Learning

8.1 Background and Contributions

We successfully pushed the frontier of 3D generative modelling to a method that can produce almost photo-realistic 3D samples while only requiring multi-view 2D images for supervision in HoloFusion [71] (chapter 7). Moving forward, we analyzed the proposed HoloFusion pipeline and found that there is scope for improving the computational efficiency, while noting that exercising frugality in the computational budget is quite important for our research agenda. Firstly, the proposed pipeline has two stages, of which, the second fine-tuning optimization stage is quite computationally expensive. And secondly, the use of feature-voxel-grids adds quite a lot of redundant computations in free-space. Interestingly, focusing more on the second phenomenon, we find that if the method could use a more compact 3D representation such as Triplanes [169, 209], then we could use a bigger first stage backbone network to predict the Triplanes, making the second finetuning stage obsolete. Thus replacing feature-voxel-grids with Triplanes could solve two problems in one go. However, the operation where we encode 2D views into an approximate 3D representation at the beginning of the HoloFusion pipeline is far from being trivial to adapt to Triplanes maintaining a robust feature flow in the pipeline. While doing this encoding for other 3D neural representations such as Hash-grids and MLPs is seemingly impossible. It turned out that the solution for 2D views to Triplane encoding that I invented was much more general, allowing for not only Hash-grids, and MLPs, but any arbitrary 3D representation that may be proposed in the future as well. In this chapter, we describe this problem and this intriguing solution proposed by us in great details.

I worked on this project almost autonomously while receiving timely advice from David, Niloy, and Andrea. Roman and Tom are also authors on this publication because I used some of the code they had developed internally at Meta.

8.2 Introduction

Encoding information from 2D views of an object into a 3D representation is crucial for generalized 3D feature extraction and learning. Such features then enable various 3D applications, including reconstruction and generation. We propose GOEmbed: Gradient Origin Embeddings that encodes input 2D images into any 3D representation, without requiring a pre-trained image feature extractor; unlike typical prior approaches in which input images are either encoded using 2D features extracted from large pre-trained models, or customized features are designed to handle different 3D representations; or worse, encoders may not yet be available for specialized 3D neural representations such as MLPs and Hash-grids. We extensively evaluate our proposed general-purpose GOEmbed under different experimental settings on the OmniObject3D benchmark. First, we evaluate how well the mechanism compares against prior encoding mechanisms on multiple 3D representations using an illustrative experiment called Plenoptic-Encoding. Second, the efficacy of the GOEmbed mechanism is further demonstrated by achieving a new SOTA FID of 22.12 on the OmniObject3D generation task using a combination of GOEmbed and DFM (Diffusion with Forward Models), which we call GOEmbedFusion. Finally, we evaluate how the GOEmbed mechanism bolsters sparse-view 3D reconstruction pipelines.

The rate of progress in 3D Computer Vision research has risen in the last decade due to increased interest in various AR (Augmented Reality), VR (Virtual Reality) and MR (Mixed Reality) applications [56, 55, 68, 215, 216]. Many 3D



Figure 8.1: We propose the **GOEmbed** (Gradient Origin **Embed**ding) mechanism that encodes source views (o^{ctxt}) and camera parameters (ϕ^{ctxt}) into arbitrary 3D Radiance-Field representations g(c,d) (sec. 8.3). We show how these generalpurpose GOEmbeddings can be used in the context of 3D DFMs (Diffusion with Forward Models) (sec. 8.5) and for sparse-view 3D reconstruction (sec. 8.6).

Computer Vision problems are newly being viewed in the light of Deep-Learning based solutions. The process of encoding the information in 2D images into deep features over the chosen 3D representation can be found in the Deep-Learning solutions to almost all these problems, for instance, consider various solutions to long-standing problems such as MVS (Multi-View Stereo) [105, 125, 126], NVS (Novel-View Synthesis) or IBR (Image Based Rendering) [31, 217, 108, 109, 218], and 3D reconstruction [219, 220, 221], as well as to the nascent 3D problems such as 3D synthesis [70, 222, 223], and 3D distillation [71, 224]. Surprisingly, despite being such a critical operation, no systematic standalone study of this 2D to 3D encoding operation exists (to the best of our knowledge).

3D scenes/assets do not have a *de-facto* data representation, and different representations are utilized depending upon the requirements of the applications. For instance, just for representing Radiance Fields of static 3D assets, various neural

data representations such as MLPs [34, 107, 33], Triplanes [67, 209], Feature-voxelgrids [63, 225], Hash-grids [66, 226], as well as non-neural ones like ReLU-Fields [64], Plenoxels [123], DVGo [65], 3DGS [227] are being utilized in various 3D applications. Interestingly, none of these is *foolproof* and each has certain prosand-cons. For instance, on the one hand NeRF MLPs can very compactly (\sim 5MB) represent extensive 3D spatial scenes, but on the other require excessively long times for training (\sim 1-2 days) and rendering (\sim 1-2 mins). Thus, given this disarray around 3D scene representations, it is a key challenge to come up with a 2D-to-3D encoding method that can: (i) generalize to arbitrary 3D representations, (ii) while being able to capture maximum information in the 3D features from the 2D images.

Existing methods of encoding can be grouped into two categories. (i) In the first category, the methods are similar to cost-volume-construction-like approaches where 2D deep features are extracted from the images, and then these 2D features are un-projected into the 3D space. On top of the deep-feature extraction network, this un-projection operation can require predicted depths using off-the-shelf depthestimation models [228, 229, 230] for 3D representations such as point-clouds [231, 102, 232]. For 3D representations such as feature-voxel-grids, per-voxel-costs in the form of variance of the per-2D-view features implicitly encodes 3D depths but requires large amount of compute-memory [126, 105]. What is further limiting is that the un-projection operation, is non-trivial and challenging to extend to specialized 3D neural representations such as MLPs and Hash-grids, and hence the approaches from this category are mostly limited to only certain 3D representations. (ii) In the second category, the methods entirely circumvent all forms of 3D inductive biases and directly inject the 2D features into the problem specific backbone networks. For example, methods like NeRFformer [76] and LRM [31] use cross-attention to directly inject 2D features into the 3D sparse-view-reconstruction backbone network. Apart from the limitation of requiring the memory-heavy cross-attention operation, given the results of our sparse-view-3D-reconstruction experiments (sec. 8.6), we hypothesize that these learned-feature-injection approaches may not be learning sufficient 3D-priors and are specializing only to the domains of training (albeit on large-scale).

To overcome the aforementioned limitations, we propose **GOEmbed**: **G**radient **O**rigin **Embed**dings (fig. 8.1) to encode the information in 2D images into any 3D representation which exists currently or which will be proposed in the future; as long as the differentiable *render* operation can be defined on it. Succinctly, the GOEmbed defines the 3D embeddings as the gradient of the mean-squared-error between renders (of the origin 3D representation) and the G.T. 2D views wrt. a predefined origin over the chosen 3D representation (fig. 8.2 and sec. 8.3). In most cases (except for MLPs due to symmetry-breaking), a simple zero-feature initialization is sufficient to define the origin. Apart from the 3D representation agnostic general-purpose applicability, our GOEmbeddings are light-weight since they do not require memory-heavy large pretrained 2D feature-extraction networks, and they try to maximize the information transfer between 2D and 3D. In summary, our contributions are:

- We propose the **GOEmbed** as a generalized encoding mechanism of 2D source views into different 3D representations (sec. 8.3, eq. 8.1).
- We propose a novel 3D diffusion pipeline by combining our GOEmbed with DFM (Diffusion with Forward Model) to achieve the state-of-the-art score on the OmniObject3D generation benchmark (sec. 8.5, eq. 8.3 8.6).
- We evaluate the efficacy of GOEmbed for extracting different 3D representations from source images in the illustrative Plenoptic Encoding experiment (sec. 8.4); and also evaluate its utility in the sparse-view 3D reconstruction task (sec. 8.6).

8.3 Method

8.3.1 GOEmbed: Gradient Origin Embeddings

Let g(c,d) represent a static 3D scene as a Radiance-Field such that c = [x,y,z]denotes the 3D coordinates of a point in the Euclidean space, $d = [\theta, \gamma]^1$ denotes

¹We deviate from the more common use of $[\theta, \phi]$ for spherical polar coordinates in favour of $[\theta, \gamma]$ to avoid confusion with ϕ^{ctxt} and ϕ^{trgt} used to denote context and target camera parameters



Figure 8.2: GOEmbed illustration. We demonstrate the mechanism here using the Triplane representation for g(c,d), but note that this can be applied to other representations as well. The GOEmbed mechanism (eq. 8.1) consists of two steps. First we render the origin ζ_0 from the context-poses ϕ^{ctxt} ; then we compute the gradient of the MSE between the renders and the source-views o^{ctxt} wrt. the origin ζ_0 which gives us the GOEmbed encodings ζ_{enc} .

the spherical polar coordinates of an outgoing direction from the point, and the function $g : \mathbb{R}^5 \to \mathbb{R}^4$ maps each 3D coordinate and a particular outgoing direction to four values (σ, R, G, B) . Here, σ represents density and [R, G, B] represents the outgoing radiance. Let $\mathcal{R}(g, \phi)$ denote the rendering *functional* that converts the Radiance-Field function into an image of a certain resolution from a certain camera viewpoint, as described by the camera parameters ϕ . We use the differentiable Emission-Absorption Volume Raymarching algorithm for rendering [34, 72]. For the encoding mechanism to be unified and general-purpose, it must have the following three properties:

- (i) It should be able to encode one or more views alike.
- (ii) It should generalize to any parameterization/realization of the function g, i.e., the encoding mechanism should be applicable irrespective of whether the Radiance-Field is represented as an MLP [34], a Hash-grid [66], a Triplane [67, 184], or a Voxel-grid [64, 123, 65].

(iii) It should try to maximize the information transfer from the 2D views to the respectively.

3D embedding.

We introduce the Gradient Origin Embeddings, where we define the encoding of the observations (2D views) as the gradient of the mean-squared error between the rendered and ground truth 2D views with respect to a predefined origin (zero vector or features) 3D representation.

Without any loss of generality, assuming that ζ are the parameters of the *g* function (i.e., the features/weights of the 3D Radiance-Field) and ζ_0 denotes the origin (zero features/weights), we define the encodings ζ_{enc} as (fig. 8.2):

$$\zeta_{\text{enc}} := GOEmbed(g, o^{\text{ctxt}}, \phi^{\text{ctxt}})$$
$$:= -\nabla_{\zeta_0} ||o^{\text{ctxt}} - \mathcal{R}(g(c, d; \zeta_0), \phi^{\text{ctxt}})||_2^2$$
(8.1)

where, o^{ctxt} and ϕ^{ctxt} are the G.T. 2D views and their camera parameters, respectively, which are to be encoded into the 3D representation ζ_{enc} . Note that the proposed encoding function GOEmbed backpropagates through the differentiable rendering functional \mathcal{R} . By construction, such an encoding satisfies the formerly stated properties (i) and (ii). Further, we minimize the loss function

$$\mathcal{L}^{\text{GOEmbed}}(o^{\text{ctxt}}, o^{\text{trgt}}) := ||o^{\text{ctxt}} - \hat{o}^{\text{ctxt}}||_2^2 + ||o^{\text{trgt}} - \hat{o}^{\text{trgt}}||_2^2$$

where, $\hat{o} = \mathcal{R}(g(c, d; \zeta_{\text{enc}}), \phi),$ (8.2)

for maximizing the information content in the encodings ζ_{enc} to satisfy property (iii). Here, ϕ^{ctxt} are the context views used for encoding while ϕ^{trgt} are some target views of the same scene but different from the source ones. Intuitively, we repurpose the backward pass of the rendering functional to encode the information in the source views o^{ctxt} into the parameters of the 3D scene representation ζ . Thus, as long as a mathematically differentiable rendering operator is possible on it, any 3D scene representation can be encoded using our GOEmbed encoder.

8.3.2 Experimental Evaluation Rubric

We evaluate the generality of GOEmbed mechanism along three axes: firstly, to measure the information transfer, we run experiments in a Plenoptic Encoding setting (sec. 8.4); secondly, we train the GOEmbedFusion model to learn a 3D generative model using 2D images to evaluate its 3D generative capability (sec. 8.5); and finally, we also evaluate the GOEmbed mechanism in a sparse-view 3D reconstruction setting (sec. 8.6).

Dataset. We perform all our experiments on the recently released OmniObject3D dataset [195] containing \sim 6K 3D scans of real world objects from daily life. The dataset contains a large-vocabulary of \sim 200 categories of daily life classes having some overlap with COCO [233]. Only for our non-forward diffusion baseline, we also use the text-captions recently released by the OmniObject3d authors at their GitHub page [234].

Metrics. For the experiments in the Plenoptic-Encoding setting (sec. 8.4), we use the standard image reconstruction metrics **PSNR**, **LPIPS** [193] and **SSIM** [235], while for the quantitative analysis in generative modeling experiments (sec. 8.5) we use the **FID** [210] and **KID** [211] metrics following prior works. And, lastly for the 3D reconstruction experiments (sec. 8.6) we again use the standard image reconstruction metrics similar to the Plenoptic-Encoding experiments. We color code all the scores as first, second, and third.

8.4 Plenoptic Encoding

To evaluate the information transfer from the source views to the encoded 3D representation, we train the standalone GOEmbed component on its own before using it in different contexts. Specifically, given the dataset $D = \{(\mathcal{I}_i^j, \phi_i^j) | i \in [1, N] \text{ and } j \in [1, C]\}$ of *N* 3D scenes where each scene contains *C* images and camera parameters, we define the Plenoptic Encoding as a mechanism which encodes *k* source views and camera parameters, of a certain 3D scene, into the representation *g* (whose parameters are ζ). The encoded scene representation should be such that the rendered views from the same source cameras, and some *l* different target cameras,



- **Figure 8.3: Plenoptic Encoding Qualitative Evaluation**. The rows MLP, Triplane and Voxel-grid show the renders of the GOEmbed encoded representations from the target-view respectively. The colour-coded columns demonstrate the effect of varying the number-of-source views (1, 2, 3, 4) used in the GOEmbed encoding. The SSO column shows the target render of the single-scene-overfitted representation while the G.T. column shows the mesh-render from the dataset (repeated for clarity).
- Table 8.1: Plenoptic Encoding Quantitative Evaluation. PSNR(↑), LPIPS(↓) and SSIM(↑) reported on three different representations of the 3D Radiance-Field g, namely, Triplanes, Voxel-Grids and MLPs. All the metrics are evaluated for target views (different from the source views) against the G.T. mesh renders from the dataset. The SSO (Single Scene Overfitting) scores denote the case of individually fitting the representations to the 3D scenes.

Method	Triplane			Voxel-Grid			MLP		
	PSNR (†)	LPIPS (\downarrow)	SSIM (†)	PSNR (†)	LPIPS (\downarrow)	SSIM (†)	PSNR (†)	LPIPS (\downarrow)	SSIM (†)
DinoV2 1 source-view	14.182	1.286	0.425	14.419	1.151	0.485	N/A	N/A	N/A
DinoV2 2 source-views	14.482	1.235	0.435	15.330	0.925	0.526	N/A	N/A	N/A
DinoV2 3 source-views	14.664	1.213	0.430	15.685	0.864	0.546	N/A	N/A	N/A
DinoV2 4 source-views	14.711	1.199	0.425	15.834	0.857	0.551	N/A	N/A	N/A
(Ours) 1 source-view	15.711	1.025	0.477	15.514	1.008	0.479	11.208	1.124	0.496
(Ours) 2 source-views	15.513	1.068	0.468	15.319	1.067	0.480	11.406	1.197	0.489
(Ours) 3 source-views	15.672	1.129	0.456	15.590	1.110	0.484	11.490	1.202	0.500
(Ours) 4 source-views	15.919	1.150	0.472	15.755	1.145	0.486	11.599	1.195	0.505
SSO	28.165	0.087	0.941	32.061	0.067	0.9582	27.382	0.119	0.918

should be as close as possible to the G.T. images \mathcal{I} , i.e., the PlenopticEncoder $PE : \mathbb{R}^{k \times h \times w \times c} \times \mathbb{R}^{k \times 4 \times 4} \to \mathbb{R}^{(k+l) \times h \times w \times c}$ should minimize the following mean squared error objective:

$$\mathcal{L}^{ ext{PE-MSE}} := \mathbb{E}_{(\mathcal{I}, \phi) \sim D} \| \mathcal{I} - PE(\mathcal{I}, \phi) \|_2^2.$$

Although this experimental setup is quite similar to typical MVS/NVS or 3D reconstruction or 3D prior learning setups, in form and essence; we note here that the plenoptic encoder *PE* is neither targeted to do Multi-View Stereo nor 3D reconstruction. To set the correct expectations, we note that the main and the only goal here is to evaluate how much information is transferred from the 2D images to the chosen 3D representation. We evaluate the GOEmbed encodings on three different 3D Radiance-Field representations, *g*, namely Triplanes [169], Feature-Voxel grids [64, 70, 123, 65], and MLPs [34] and compare them to cost-volume like approaches where possible.

All the evaluations in Table 8.1 are done on 100 randomly chosen *test* scenes from the OmniObject3D dataset on 1, 2, 3, and 4, number of source views. Although the loss is optimized on both source and target camera views, the reported scores are for *target-views* only, since we are interested in measuring the "3D" encoding ability, as opposed to overfitting the source views into the representation. If we consider only the source-view metrics, there is a possibility of the degenerate solution where the encoder copies all the source views as-it-is disregarding any 3D structure. Also, all the scores are computed against the renders of the G.T. meshes, but since some quality can be lost by the choice of the representation itself, the SSO (Single-Scene-Overfitting) version, where we fit the representation directly to the scene, is also provided for a more grounded comparison.

Table 8.1 summarizes the scores obtained in this setup of experiments. We compare against cost-volume construction approach using DinoV2 [127], where we first un-project the per-view image features into a feature-voxel grid and then accumulate the per-view features into the cost-volume similar to StereoMachines [125] and many others [126, 70, 109, 224, 105]. We further splat the voxel-features into the mutually orthogonal planes in case of the Triplanes based cost-volume baseline. As apparent from the table, GOEmbed outperforms the DinoV2 cost-



Figure 8.4: 3D Generation Qualitative Evaluation. 3D samples generated by our GOEmbedFusion compared to the prior GAN, and Diffusion based baselines.

volume-approach for Triplanes and comes very close to the Voxel-Grids one. Note that the cost-volume approach is not trivial to come-up with for MLPs, whereas our GOEmbed can handle this representation by design and obtains formidable scores in this setup. Apart from this advantage, GOEmbed only has learnable parameters (\sim 17K) as part of the renderer functional \mathcal{R} (usually an MLP), while the DinoV2 has excess of \sim 1B parameters, albeit pretrained. Finally, what is most surprising is that our light-weight GOEmbed approach is actually able to obtain almost 50% of the SSO scores which is the practically achievable performance for the chosen 3D representations (last row of table 8.1). Figure 8.3 illustrates this phenomenon visually.

8.5 3D Generation

Although prior works like RenderDiffusion [214] and HoloDiffusion [70] were already proposed to train 3D diffusion models using only 2D images, recently, the notable work of Tewari et al. [222] proposed a unified mathematical framework for the *stochastic-inverse* setting of generative modeling where we only have access to partial observations of the underlying ground-truth signals, but never the underlying signals themselves; in which 3D inverse graphics is a special case. Despite this, the proposed vanilla realization of the DFM framework for 3D generative modeling has certain limitations; first, it is only specific to the feature-point-cloud 3D representation and does not generalize to other 3D representations; second, it cannot generate purely unconditional 3D samples since it is a 2D view conditioned 3D diffusion model; and third, it requires a computationally expensive auto-regressive process for sampling the underlying 3D scenes.

We propose a novel realization of the DFM framework called GOEmbedFusion where our proposed GOEmbed drives the diffusion with forward model training pipeline; while overcoming all the limitations as mentioned earlier. Our unobserved samples here correspond to the ζ parameters since we are interested in modeling the generative probability distribution over the 3D Radiance-Fields. A single forward pass through our proposed GOEmbedFusion pipeline is defined through the following equations:

$$\zeta_{\text{enc}} := GOEmbed(g, o^{\text{ctxt}}, \phi^{\text{ctxt}})$$
(8.3)

$$\hat{\zeta}_0 := \mathcal{D}_{\theta}(\zeta_T, T; \zeta_{\text{enc}}) \tag{8.4}$$

$$\hat{\zeta}_{t} \sim q(\hat{\zeta}_{t}|\hat{\zeta}_{t-1}) = q(\hat{\zeta}_{0}) \prod_{k=1}^{t} q(\hat{\zeta}_{k}|\hat{\zeta}_{k-1})$$
(8.5)

$$\hat{o}^{\text{trgt}} := \mathcal{R}(g(c,d;\mathcal{D}_{\theta}(\hat{\zeta}_t,t;\zeta_{\text{enc}})),\phi^{\text{trgt}}),$$
(8.6)

where *T* corresponds to the highest timestep and $\zeta_T \sim \mathcal{N}(0, I)$ is a sample of the standard Gaussian noise distribution. Equation 8.3 first encodes the context observations and their parameters ($o^{\text{ctxt}}, \phi^{\text{ctxt}}$) into the GOEmbed encoding ζ_{enc} using eq. 8.1. Then, eq. 8.4 uses the denoiser network to predict a pseudo-deterministic estimate of the clean 3D scene $\hat{\zeta}_0$ conditioned on the GOEmbed encodings ζ_{enc} . While the noise can impart some degree of stochasticity to the output, the network can, in theory, completely ignore the noise. We let the network learn what to do based on the training and the data complexity, and hence call this step "pseudo"-deterministic. This is followed by obtaining a noisy version of the 3D scene $\hat{\zeta}_t$ through standard forward diffusion corruption (eq. 8.5). We estimate the process of drawing from the distribution $q(\hat{\zeta}_t | \hat{\zeta}_{t-1})$ using the standard DDPM [80] closed-form equation $\hat{\zeta}_t = \sqrt{\alpha_t} \hat{\zeta}_0 + \sqrt{1 - \alpha_t} \varepsilon$, where ε is pure Gaussian noise and α_t denotes the schedule of diffusion. And finally, we predict any target observations by rendering the output of the denoiser network \mathcal{D}_{θ} for $\hat{\zeta}_t$ conditioned on the GOEmbed encodings ζ_{enc} and the timestep *t* using eq. 8.6.

We note here that the network \mathcal{D}_{θ} operates in the x_start diffusion formulation in contrast to the popular epsilon formulation. We require a few different loss functions to be able to train this pipeline end-to-end such that each component does what it is supposed to:

$$\begin{split} \mathcal{L}^{\text{GOEmbedFusion}} &:= \mathbb{E}_{t,o^{\text{trgt}}} \| o^{\text{trgt}} - \hat{o}^{\text{trgt}} \|_2^2 \\ \mathcal{L}^{\text{PSE-DET}} &:= \mathbb{E}_{t,o^{\text{trgt}}} \| o^{\text{trgt}} - \mathcal{R}(g(c,d;\hat{\zeta}_0)), \phi^{\text{trgt}}) \|_2^2 \end{split}$$

The final objective is simply the sum of the three loss functions as,

$$\mathcal{L}^{\text{total}} := \mathcal{L}^{\text{GOEmbedFusion}} + \mathcal{L}^{\text{PSE-DET}} + \mathcal{L}^{\text{GOEmbed}}.$$

The loss function $\mathcal{L}^{\text{GOEmbed}}$ is required to maximize the information content in the GOEmbed encodings. In contrast, the $\mathcal{L}^{\text{PSE-DET}}$ tries to maximize the reconstruction quality of the pseudo-deterministic output $\hat{\zeta}_0$. The $\mathcal{L}^{\text{GOEmbedFusion}}$ loss function actually trains the GOEmbedFusion pipeline. As apparent, the GOEmbed mechanism enables the GOEmbedFusion model to use any parameterization of the *g* function as long as it allows differentiable rendering and can define a zero-origin over its parameters ζ . Empirically, we found that the two-step bootstrapped-denoising as done in eqn. 8.1 and eqn. 8.6, similar to prior works [70, 71], is crucial to correctly train this diffusion pipeline in the forward setting. In practice, we train the model using the classifier free guidance training scheme [236], where we dropout the GOEmbed conditional sampling. Lastly, samples can be generated by iterative denoising using the trained model \mathcal{D}_{θ} either unconditionally or conditionally using the

GOEmbed encodings of certain context observations. Here, we can directly sample (denoise) in the space of the unobserved underlying data samples ζ , eliminating the need for the expensive auto-regressive fusion required by the vanilla realization.

Table 8.2: 3D Generation Quantitative Evaluation. FID(↓) and KID(↓) scores on the OmniObject3D dataset comparing our GOEmbedFusion with GAN baselines EG3D [169], and GET3D[172]; with the non-forward diffusion baselines DiffRF[185], DiffTF[237], and Our non-forward diffusion baseline; and, with the DFM (Diffusion with Forward Model) [222].

Method	$\mathrm{FID}(\downarrow)$	$\mathrm{KID}(\downarrow)$
EG3D [169]	41.56	1.0
GET3D [172]	49.41	1.5
DiffRF [185]	147.59	8.8
DiffTF [237]	25.36	0.8
DFM [222]	73.51	3.8
Ours non-forward	119.67	8.0
GOEmbedFusion (Ours)	22.12	0.6

As shown in table 8.2, our GOEmbedFusion sets the new state-of-the-art FID and KID scores on the OmniObject3D dataset in comparison to the prior stateof-the-art DiffTF [237], our implementation of a DiffTF like baseline called Ours non-forward, the DFM [222] model, the DiffRF [185] model, and the prior GANs EG3D [169] and GET3D [172]. We note that our improvements are not only limited to the quality of generation, but also to the benefits provided by our GOEmbedFusion formulation over the vanilla realisation of the DFM (row 5 table 8.2). While DiffTF [237] uses many architectural modifications and other tricks specific to Triplanes and the OmniObject3D dataset, we only use the base DiT [238] architecture with *no* modifications as our backbone denoiser network \mathcal{D}_{θ} . Also our proposed GOEmbed-Fusion training pipeline is a diffusion with forward model, and hence can be trained only using 2D images, unlike DiffTF which first fits ground-truth 3D Triplanes and then trains the diffusion model. Our qualitative samples in figure 8.4 further support these arguments.

8.6 Sparse-View 3D Reconstruction

The Plenoptic Encoding experiments (sec. 8.4) are in an illustrative setup and provide insights into how well our propsed GOEmbed can transfer information from 2D images into various different 3D representations such as Triplanes, MLPs and Voxel-Grids. In this section, we aim to evaluate the utility of the GOEmbeddings in a practical application setup. Thus, although the mathematical experimental setup is similar to the Plenoptic Encoding one, here we input the obtained GOEmbeddings to a backbone sparse-view-3D-reconstruction network. Specifically, we use the DiT based transformer network, which is 12-layers wide as the reconstruction backbone. The end-to-end pipeline is supervised with exactly the same losses as that of the Plenoptic Encoding setup. Intuitively, our GOEmbed reconstruction model replaces the input learned triplane positional encodings with our GOEmbeddings and removes the cross-attention layers from the LRM (Large Reconstruction Model) architecture. Additionally, we also evaluate the reconstruction capability of our GOEmbedFusion model by running the pipeline only till the "pseudo"-deterministic output stage (i.e., eqn. 8.4).

We compare this transformer-based reconstruction setup of ours to the most recent baseline of LRM (Large Reconstruction Model) [31], which is also based on the transformer architecture. Since LRM's code has not been published, we implement this baseline in our code-base as close to the paper-description as possible, for a fair comparison. Specifically, we train two versions of the LRM, the first one which is 16-layers wide (the base published model), and second smaller version which is 6-layers wide. We introduced the 6-layers version since we found the base-model to be overfitting to the OmniObject3D training subset, which is much smaller in scale than the dataset on which LRM is trained. All the learned network baselines are evaluated on the ~ 200 test scenes of the OmniObject3D dataset while the Triplane-SSO (from the Plenoptic Encoding section 8.4) is evaluated on 100 scenes. These 100 SSO scenes form a proper subset of the test-set, so the comparison here is fair. Please check the supplemental for more details of our GOEmbed reconstruction architecture.

Method PSNR (\uparrow) LPIPS (\downarrow) SSIM (\uparrow)								
LRM (Our) 6 layer	23.788	0.119	0.827					
LRM (Our) 16 layer	23.247	0.121	0.813					
GOEmbed recon	27.650	0.109	0.900					
GOEmbedFusion (PSE-DET)	26.447	0.119	0.890					
Triplane SSO	28.165	0.087	0.941					

Table 8.3: 3D reconstruction Quantitative Evaluation. PSNR(↑), LPIPS(↓) and SSIM(↓) of our GOEmbed reconstruction model, and GOEmbedFusion's "pseudo"-deterministic 3D reconstruction output compared to LRM baselines. We again include the SSO (Single Scene Overfitting) here for comparison.

As summarized in the table 8.3, both our GOEmbed reconstruction model as well as the diffusion based GOEmbedFusion model outperforms the LRM baselines. Also, our reconstruction model gets quite close to the practical performance limit as set by the Triplane-SSO experiment. Thus, given these results, we can assert that our proposed GOEmbeddings generalise to larger datasets like OmniObject3D and can strengthen the sparse-view-3D-reconstruction pipelines.

8.7 Limitations

Although our framework applies theoretically to any arbitrary 3D Radiance-Field representations, its use in the 3D generative modeling is restricted by the representation's compatibility with existing denoiser network architectures. We believe that Transformers [49] get close to being universally applicable, but it still remains a challenge to adapt certain 3D representations such as Hash-grids [66] as input to Transformers. Nevertheless, we believe that our proposed GOEmbed mechanism makes a substantial research stride and will inspire further interesting applications and theoretical insights. Apart from this, upon close qualitative examination, we find that the samples generated using our GOEmbedFusion model have some peculiar checkerboard artifacts. Similar to the findings of the recent Denoising-ViT [239], we hypothesize that these artifacts in our model are also because of the positional encodings in the DiT architecture. Finding the exact reason for these artifacts and getting rid of them constitutes a future direction to be pursued.

Although our proposed method primarily contributes to general-purpose 3D

feature extraction and learning, as shown in our 3D generation experiments, our method could be applied in the context of the generative modeling of real-captured or synthetic 3D assets. Hence, similar to the case of 2D generative models, our proposed GOEmbedFusion model is also prone to the misuse of the synthetically generated media. We note that there is a potential for our method to be used in the creation of fake 3D-consistent videos.

8.8 Summary

We presented the GOEmbed as a general-purpose mechanism for encoding the information in 2D images into arbitrary 3D scene representations and evaluated its information transfer ability with the Plenoptic Encoding experiments. We show that the encodings can be applied to Triplanes, Voxel-grids and MLPs, but note that exploring these in the context of other popular 3D representations (e.g., meshes, point clouds) forms scope for future work. The GOEmbeddings find a practical use in the context of improving the framework of DFM models, which we demostrate through the 3D generation experiments on the OmniObject3D benchmark; and in the context of sparse-view 3D reconstruction as well.

Chapter 9

Conclusions

9.1 Summary

In this thesis, I have tried to pave the way towards achieving computationally efficient, photorealistic, and scalable 3D generative models. In this section, I present a discussion of how the proposed methods in this research journey contribute specifically to each of the characteristics of the ultimate 3D generative models that we wish to achieve. While the first project, ReLU-Fields (chapter 4), showed a key insight on what is the bare-minimum required change to the traditional 3D voxel-grids that allows them to match the quality of NeRFs, the rest of the projects that followed, all focussed on 3D generative modelling, and contributed to the ultimate vision of achieving foundational 3D generative models.

Computational Efficiency is certainly the most important property required in 3D Generative Models. Perhaps the GPUs in the future will have Terabytes of on-board memory and might allow processing of very high-resolution 3D voxel grids, but even then the Neural Networks won't be scalable if they wasted computations in free-space. Computational efficiency has served as an important guiding principle when designing methods for 3D generative modelling in this thesis. HoloDiffusion [70] (chapter 6) circumvents the need for the redundant step of fitting 3D assets given multi-view 2D images in order to train a 3D diffusion model. Typically, fitting a feature-voxel-grid with sufficient quality on a single 3D scene, blocks one GPU with minimum of 32GB on-board memory for 2-3 hours. Thus constructing a corpus of

9.1. Summary

50K scenes, would require roughly \sim 5K GPU hours. Unlike methods such as RODIN [184] and DiffRF [185] which perform this fitting stage to construct a dataset of 3D representations prior to training the diffusion model, our proposed HoloDiffusion saves up on crucial computational budget by training the 3D diffusion model directly on multi-view 2D images. Later when trying to scale up our HoloFusion [71] (chapter 8) pipeline to larger datasets, we investigate what can make the pipeline more computationally efficient to arrive at the pipeline of GOEmbedFusion (chapter 8). Our proposed GOEmbed operation allows us to encode multi-view 2D images into arbitrary 3D scene representation such as Triplanes which are more efficient than the previously used feature-voxel-grids in HoloDiffusion and HoloFusion.

Photorealism. The ideal 3D Generative Model should be able to produce 3D samples that are indistinguishable from the reality. The 3D objects in real life are seen by our eyes due to the interaction of the light with the objects. This gives rise to the notion of photorealism as perceived by the human brain. How can we achieve photorealism in the synthetic samples generated by our 3D generative models? The answer is simple yet extremely challenging. We need to capture as much real-life 3D data as possible and then use it for training the 3D generative models. Our proposed HoloDiffusion [70] (chapter 6) makes a very important contribution towards this goal by proposing a method that can directly train 3D diffusion models using real-captured multi-view imagery, since multi-view images remain to be the most cost-effective solution for capturing 3D assets in real life. With a hybrid 3D-2D extension of the HoloDiffusion pipeline, we show in HoloFusion [71] (chapter 7) that the visual fidelity of the generated samples can be taken to the level of photorealism on real-captured 3D datasets. I hypothesize that a large enough 3D generative model trained on billions of real-captured 3D scenes will be able to understand the physics behind the lightinteractions, and thus be able to not-only generate photorealistic 3D assets, but also allow for physically accurate manipulation of lighting.

Scalability. As more and more 3D data keeps getting accumulated, the proposed methods of 3D Generative Modelling should exhibit linear scale-curves. I.e. proportionately bigger Neural Network should be able to train on bigger data with more

compute. However, for 3D data, this is not so straight-forward as 2D images, since more data will mean more disarray around the 3D representations. Hence, the only solution for achieving scalability in 3D generative modelling is for the research works to encompass continual empirical growth in the benchmark dataset sizes. As the sizes grow, new challenges will keep getting uncovered and will have to be overcome. We demonstrate exactly this approach towards scalability through our research projects. 3inGAN [69] (chapter 5) started by training on a single 3D scene, followed by HoloDiffusion [70] and HoloFusion [71] (chapters 6 and 7) which increased the datasets size to a few 1000 samples, till the most recent proposal of GOEmbedFusion (chapter 8) which was trained on the OmniObject3D dataset containing many more diverse classes and total samples more than Co3D individual classes. Also from some of the internal experiments we were able to establish successful training of GOEmbed on >100K data samples using a backbone architecture similar to LRM [31]. In a nutshell, we need to keep increasing the sizes of the benchmarks gradually while simultaneously exploring different 3D representations to keep improving the scalability properties of the 3D generative modelling methods.

9.2 Insights

This has been a long journey starting from absolute scratch (some prior knowledge of generative modelling, and no background in 3D applications) till reaching a method capable of training a 3D diffusion based generative model of any arbitrary 3D representation using only 2D images for supervision (GOEmbed chapter 8). I have learned a lot from the successes and the failures on this journey, and so, I now present the main highlights distilled from it as follows.

Don't discard old methods, review them in new light. Most of the times when we were stuck on a problem, we found its solution to be rooted in a well developed traditional method. The first and the best example for this is ReLU-Fields. While the research community was absolutely mesmerized by the amazing quality of the reconstructions produced by NeRFs, who would have thought that a simple "ReLU" non-linearity is all you need to bring the traditional linear voxel grids to the quality of NeRFs. While working on HoloDiffusion, for encoding 2D images into 3D voxel grids, we had to resort to the feature-cost-volume construction module which has been around for at least a decade. These are only the most relatable examples that I highlighted here, but I will certainly note that in almost all the projects, we do end-up applying a method which has very old roots. The point I am trying to make is two-fold. Firstly, instead of reinventing the wheel, it's quite important to understand why we stand where we stand and then build up on the knowledge base constructed by the previous generation of researchers. But secondly, which I think is the most important, is to not rely on huge parametric-networks for solving all problems. The solutions should exhibit computational efficiency and try to apply strong inductive biases in the proposed methods.

GANs or Diffusion Models? This debate has not been concluded yet, especially since the work GigaGAN [15] has been released. However, I assert that diffusion models remain preferable for training 3D generative models compared to GANs, since 3D data comes with an inherent scale issue and thus leaves less scope for hyperparameter tuning without which GAN training is impossible. Also from my experience, the hyperparameter settings for training GANs on 3D data do not transfer from one domain to the other, and thus significant time and compute resources are required to successfully train the GANs. Whereas, Diffusion models behave as stably as any other supervised Machine Learning model, and thus are more preferable for 3D generative modelling. This is of course unless a break-through idea challenges the unstability issue of GANs, making them as easy-to-use as Diffusion Models. To go a step further in this debate, from my experience, I find that GANs are really good at covering fine-peaks in the generative distributions, while Diffusion Models are very good at capturing the holistic modes of the distributions. Consider the adversarial LPIPS used by Stable-Diffusion [12] which brings the high-frequency details in the AutoEncoder reconstructions. Upon closer observation, I find that the results of 3inGAN have a lot of high-frequency details in them, almost causing the geometries to shatter in order to produce them, while the 3inFusion results are quite smooth. As a general recipe, I suggest to first train diffusion model to learn the coarse structure of the generative distribution, and then train a GAN to refine the produced samples and bring the realism in them. Also, the best strategy for training GANs successfully in my opinion is to first pre-train the GAN stage with a simpler loss like Mean-Squared error, and then kick in the discriminator loss. This strategy always works allowing for the discriminator loss to cover the last-mile in the results. In nutshell, Diffusion for obtaining 99% of the results and GANs for obtaining the last 1%.

Real captured data or Synthetic? This is a question that requires more thorough scientific research while progressing on the path of achieving 3D generative models. However, from my experience, I think there is a clear distinction between these two settings. For synthetic scenes, the main challenge, as well as the goal, is to obtain very high quality of visual fidelity while also allowing the users of the generative model to exert high level of control on the generation process. Whereas for the real-captures the main challenge is to train using *in-the-wild* data. For the synthetic case, the datasets of large-scale textured meshes have started to rise; for instance the Objaverse [24], the ObjaverseXL [240], and the dataset made available by Shutterstock. Similar to these, I would expect many other gaming-studios and vfx-studios to also try to democratize their 3D assets to aid 3D generative modelling. While, in case of the real setting, 360° captures of real objects are much more expensive and 3D reconstruction quality is heavily dependent on Structure-from-Motion [75]. I would hypothesize that generative models on synthetic data would not-only make more scientific and social impact, but also aid the process of the 3D generative modelling on real-data. Thus, at least for the immediate term, I will focus more on the generative modelling of synthetic 3D data after my PhD.

The devil is in the detail. Lastly, I would like to highlight the importance of tiny details that get missed while looking at the larger-picture of the proposed methods. For instance, training NeRFs is impossible with ReLU-MLPs if the training density noise is not added; our proposed ReLU-Fields pin-points exactly such a detail that makes the most impact in terms of reconstruction quality; HoloDiffusion will not work without the bootstrapped training; and many more. It is not easy to predict

which details of a proposed method are how important; and it is likely not going to be possible to always run the code of the proposed method many times to get exactly the results that the authors were able to achieve. I have tried to highlight these deviled-details as much as possible in this thesis, but it is still hard to meticulously enlist all such details. The bottom-line is that we need to explore, analyze, ablate, and understand proposed methods on a very deeper level in order to make sustainable progress on this path to 3D generative models.

9.3 What next?

In this final section, I discuss some directions that I would like to pursue further after the PhD. I also present some thoughts about the future of 3D generative modelling as follows:

9.3.1 3D Meshes are important

3D Meshes are quite important for two reasons. Firstly since most of the offline and online renderers use 3D Mesh based assets to represent the geometries, most of the available G.T. synthetic 3D data is going to be in the form of meshes. And secondly, in spite of very promising new 3D representations such as Gaussian Splats [227], most of the rendering machinery (software and hardware) is customised to meshes. Also the controllability and compatibility with physics-engines is a major contributing factor for the popularity of meshes. Thus, 3D generative modelling methods that take as input 3D meshes and that produce 3D Meshes as output are going to be important. As mentioned earlier in the introduction (chapter 1), optimization based 3D reconstruction of surface meshes is hard, thus methods such as Flexicubes [241] are going to play a key role in the immediate future, although this problem is somewhat orthogonal to 3D generative modelling. Flexicube grids are readily compatible with the existing backbone neural network architectures such as UNets and Transformers, thus a method that can filter Meshes into Flexicubes without point-based optimization is an interesting and quite needed future direction to explore. Such a method would boslter 3D generative pipelines to produce 3D Meshes directly as output. The resolution of the Meshes is restricted by Flexicube

grid-size, thus methods that can represent hierachy of coarse-to-fine details of the G.T. high-resolution meshes in a manner that is amenable for 3D generative modelling is also another prudent future direction to pursue.

9.3.2 Scale is directly proportional to impact.

Lastly I would like to highlight challenges and opportunities that lie ahead when focussing on scaling up the 3D generative models. We can all agree that an impactful breakthrough 3D generative model will be trained on at-least a billion 3D samples. The current methods have only been scaled to 100K samples successfully, and almost all of these seem to be hitting the glass ceiling when trying to scale beyond this size. I think there are various aspects that the current methods are missing which need to be incorporated when moving ahead in the scale axis.

- Sparsity: The key difference between 3D and 2D data is that 3D data is inherently sparse. Almost all of the 2D images have much more information per-capita, while the 3D meshes only contain information on the surfaces, which is a very small subset of the 3D domain. This makes it very challenging to choose a particular 3D representation from the ones that we have available. For instance, Triplanes are quite compact compared to voxel-grids, but even the projection planes of Triplanes have sufficiently large empty backgrounds. Hash-grids are supposedly more information rich than Triplanes, but they do not have spatial structure making them incompatible with the existing 3D neural network architectures. Thus, the two possible interesting future directions are: 1. innovate neural network architectures to operate on Hashgrids, and 2. invent 3D spatial representation that is even more compact than Triplanes. Nevertheless, a principled study of the differences between 2D and 3D data is required to go forward.
- 2. Latent-compression: One of the main innovations that allowed Stable-Diffusion to scale to large-datasets is the latent-compression. This idea of operating in the latent space while delegating the coding-and-decoding of raw information in the data-space to a dedicated AutoEncoder has been applied by

many others, including the latest video diffusion model from OpenAI titled SORA. I hypothesize that this latent-compression operation while considering the inherent sparsity of the 3D data will be of prime importance in scaling up the 3D generative models.

- 3. **Quality of text:** The quality of the text-captions is certainly an overlooked aspect in the present state-of-the-art 3D generative models. It has been sufficiently proven by prior works that the text-captions play an important role in training large-scale 2D generative models, and thus improving the coherence of the text-captions of the available 3D data should be prioritised.
- 4. Coarse-to-fine: Although latent-diffusion and cascaded-diffusion models perform coarse-to-fine generation implicitly, various new works such as VAR [22] are bringing back the coarse-to-fine generation characteristic of the state-of-the-art GAN models in 2D. Even though this is a hard research direction for 3D Meshes, I believe that coarse-to-fine is not-only important for scaling up the 3D generative models, but also for availing various control axes over the generation process.
- 5. **Simplicity:** Finally, I would like to draw attention to the principle of Occam's Razor, emphasizing on importance of simplicity in the proposed approaches. Many works are proposing variations of multi-stage pipelines comprising first going from text-to-2D then from 2D to multi-view 2D and then from multi-view 2D to 3D as approaches for text-to-3D generation. Apart from being complicated, such large pipelines have many moving parts and are prone to be bottle-necked by certain points of failure. Although such pipelines could yield low-hanging fruits, In my humble opinion, they are not going to be viable in the long run. The methods for 3D generative modelling should think about the holistic research on this path and should have as few stages as possible.

This concludes my Doctoral Thesis, and I sincerely hope that this not-only contributes to further research, but also inspires new readers to pursue this path.

Bibliography

- [1] Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, Merlin Nimier-David, Delio Vicini, Tizian Zeltner, Baptiste Nicolet, Miguel Crespo, Vincent Leroy, and Ziyi Zhang. Mitsuba 3 renderer, 2022. https://mitsuba.readthedocs.io/en/latest/src/inverse*rendering/radiance_fieldreconstruction.html*
- [2] Steven Harrington. Computer graphics: a programming approach. McGraw-Hill, Inc., 1987.
- [3] David F Rogers and James Alan Adams. *Mathematical elements for computer graphics*. McGraw-Hill, Inc., 1989.
- [4] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically based rendering: From theory to implementation*. MIT Press, 2023.
- [5] Adam Marrs, Peter Shirley, and Ingo Wald. *Ray tracing Gems II: next generation real-time rendering with DXR, Vulkan, and OptiX.* Springer Nature, 2021.
- [6] Tomas Akenine-Moller, Eric Haines, and Naty Hoffman. *Real-time rendering*. AK Peters/crc Press, 2019.
- [7] Wikipedia. Film industry Wikipedia, the free encyclopedia. http://en. wikipedia.org/w/index.php?title=Film%20industry&oldid=1185181785, 2023. [Online; accessed 16-November-2023].
- [8] Wikipedia. Video game industry Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Video%20game%
 20industry&oldid=1184994563, 2023. [Online; accessed 16-November-2023].

- [9] Kip Thorne. The science of Interstellar. WW Norton & Company, 2014.
- [10] James Cameron, Stephen E Rivkin, and John Refoua. Avatar. Twentieth Century Fox Home Entertainment, 2010.
- [11] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks. *Google research blog*, 20(14):5, 2015.
- [12] 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.
- [13] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. arXiv preprint arXiv:2204.06125, 2022.
- [14] 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.
- [15] Minguk Kang, Jun-Yan Zhu, Richard Zhang, Jaesik Park, Eli Shechtman, Sylvain Paris, and Taesung Park. Scaling up gans for text-to-image synthesis. In *Proceedings* of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 10124–10134, 2023.
- [16] OpenAI. Video generation models as world simulators. https://openai.com/ index/video-generation-models-as-world-simulators, 2024. [Online; accessed 5-May-2024].
- [17] Xuanyi Li, Daquan Zhou, Chenxu Zhang, Shaodong Wei, Qibin Hou, and Ming-Ming Cheng. Sora generates videos with stunning geometrical consistency. *arXiv* preprint arXiv:2402.17403, 2024.

- [18] Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18392–18402, 2023.
- [19] 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*, pages 3836–3847, 2023.
- [20] Dave Zhenyu Chen, Yawar Siddiqui, Hsin-Ying Lee, Sergey Tulyakov, and Matthias Nießner. Text2tex: Text-driven texture synthesis via diffusion models. *arXiv preprint arXiv:2303.11396*, 2023.
- [21] Zhisheng Xiao, Karsten Kreis, and Arash Vahdat. Tackling the generative learning trilemma with denoising diffusion gans. *arXiv preprint arXiv:2112.07804*, 2021.
- [22] Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. arXiv preprint arXiv:2404.02905, 2024.
- [23] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.
- [24] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi.
 Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153, 2023.
- [25] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. *Advances in Neural Information Processing Systems*, 36, 2024.
- [26] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Textto-3d using 2d diffusion. arXiv preprint arXiv:2209.14988, 2022.

Bibliography

- [27] Jiaxiang Tang. Stable-dreamfusion: Text-to-3d with stable-diffusion, 2022. https://github.com/ashawkey/stable-dreamfusion.
- [28] Jiahao Li, Hao Tan, Kai Zhang, Zexiang Xu, Fujun Luan, Yinghao Xu, Yicong Hong, Kalyan Sunkavalli, Greg Shakhnarovich, and Sai Bi. Instant3d: Fast textto-3d with sparse-view generation and large reconstruction model. *arXiv preprint arXiv:2311.06214*, 2023.
- [29] Yuan Liu, Cheng Lin, Zijiao Zeng, Xiaoxiao Long, Lingjie Liu, Taku Komura, and Wenping Wang. Syncdreamer: Generating multiview-consistent images from a single-view image. arXiv preprint arXiv:2309.03453, 2023.
- [30] Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and Xiao Yang. Mvdream: Multi-view diffusion for 3d generation. *arXiv preprint arXiv:2308.16512*, 2023.
- [31] Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. Lrm: Large reconstruction model for single image to 3d. arXiv preprint arXiv:2311.04400, 2023.
- [32] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019.
- [33] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. Advances in Neural Information Processing Systems, 32, 2019.
- [34] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In ECCV, pages 405–421, 2020.
- [35] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions, 2020.

- [36] Michael Sipser. Introduction to the theory of computation. *ACM Sigact News*, 27(1):27–29, 1996.
- [37] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.
- [38] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.
- [39] AG Ivakhnenko and GA Ivakhnenko. The review of problems solvable by algorithms of the group method of data handling (gmdh). *Pattern recognition and image analysis c/c of raspoznavaniye obrazov i analiz izobrazhenii*, 5:527–535, 1995.
- [40] Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.
- [41] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.
- [42] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278– 2324, 1998.
- [43] 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.
- [44] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing* systems, 25, 2012.
- [45] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.
- [46] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going

Bibliography

deeper with convolutions. In *Proceedings of the IEEE conference on computer vision* and pattern recognition, pages 1–9, 2015.

- [47] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.
- [48] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning internal representations by error propagation, 1985.
- [49] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.
- [50] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv* preprint arXiv:1410.5401, 2014.
- [51] Grant Sanderson. But what is a gpt? visual intro to transformers chapter 5, deep learning. https://www.youtube.com/watch?v=wjZofJX0v4M, 2024. [Online; accessed 8th-April-2024].
- [52] Grant Sanderson. Visualizing attention, a transformer's heart chapter 6, deep learning. https://www.youtube.com/watch?v=eMlx5fFNoYc, 2024. [Online; accessed 8th-April-2024].
- [53] Herbert E. Robbins. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.
- [54] Brent Burley and Walt Disney Animation Studios. Physically-based shading at disney. In Acm Siggraph, volume 2012, pages 1–7. vol. 2012, 2012.
- [55] Ayush Tewari, Justus Thies, Ben Mildenhall, Pratul Srinivasan, Edgar Tretschk, Wang Yifan, Christoph Lassner, Vincent Sitzmann, Ricardo Martin-Brualla, Stephen Lombardi, et al. Advances in neural rendering. In *Computer Graphics Forum*, volume 41, pages 703–735. Wiley Online Library, 2022.

- [56] Ayush Tewari, Ohad Fried, Justus Thies, Vincent Sitzmann, Stephen Lombardi, Kalyan Sunkavalli, Ricardo Martin-Brualla, Tomas Simon, Jason Saragih, Matthias Nießner, et al. State of the art on neural rendering. In *Comput. Graph. Forum*, volume 39, pages 701–727, 2020.
- [57] Baptiste Nicolet, Alec Jacobson, and Wenzel Jakob. Large steps in inverse rendering of geometry. ACM Transactions on Graphics (TOG), 40(6):1–13, 2021.
- [58] Delio Vicini, Sébastien Speierer, and Wenzel Jakob. Differentiable signed distance function rendering. ACM Transactions on Graphics (TOG), 41(4):1–18, 2022.
- [59] Wenzheng Chen, Joey Litalien, Jun Gao, Zian Wang, Clement Fuji Tsang, Sameh Khamis, Or Litany, and Sanja Fidler. Dib-r++: learning to predict lighting and material with a hybrid differentiable renderer. Advances in Neural Information Processing Systems, 34:22834–22848, 2021.
- [60] Shichen Liu, Weikai Chen, Tianye Li, and Hao Li. Soft rasterizer: Differentiable rendering for unsupervised single-view mesh reconstruction. *arXiv preprint arXiv:1901.05567*, 2019.
- [61] Xiuming Zhang, Pratul P Srinivasan, Boyang Deng, Paul Debevec, William T Freeman, and Jonathan T Barron. Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. ACM Transactions on Graphics (ToG), 40(6):1–18, 2021.
- [62] Haian Jin, Isabella Liu, Peijia Xu, Xiaoshuai Zhang, Songfang Han, Sai Bi, Xiaowei Zhou, Zexiang Xu, and Hao Su. Tensoir: Tensorial inverse rendering. In *Proceedings* of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 165–174, 2023.
- [63] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. arXiv preprint arXiv:1906.07751, 2019.

- [64] Animesh Karnewar, Tobias Ritschel, Oliver Wang, and Niloy J. Mitra. ReLU fields: The little non-linearity that could. In *Proc. of SIGGRAPH*, volume 41, pages 13:1–13:8, 2022.
- [65] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction, 2021.
- [66] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *arXiv:2201.05989*, January 2022.
- [67] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3d generative adversarial networks, 2021.
- [68] 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.
- [69] Animesh Karnewar, Oliver Wang, Tobias Ritschel, and Niloy J Mitra. 3ingan: Learning a 3d generative model from images of a self-similar scene. In 2022 International Conference on 3D Vision (3DV), pages 342–352. IEEE, 2022.
- [70] 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.
- [71] Animesh Karnewar, Niloy J Mitra, Andrea Vedaldi, and David Novotny. Holofusion: Towards photo-realistic 3d generative modeling. *arXiv preprint arXiv:2308.14244*, 2023.

- [72] Nelson Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995.
- [73] James T Kajiya and Brian P Von Herzen. Ray tracing volume densities. ACM SIGGRAPH computer graphics, 18(3):165–174, 1984.
- [74] Frank Dellaert and Lin Yen-Chen. Neural volume rendering: Nerf and beyond. arXiv preprint arXiv:2101.05204, 2020.
- [75] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *IEEE CVPR*, pages 4104–4113, 2016.
- [76] Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10901–10911, 2021.
- [77] Sam Bond-Taylor, Adam Leach, Yang Long, and Chris G Willcocks. Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models. *IEEE transactions on pattern analysis and machine intelligence*, 2021.
- [78] 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.
- [79] Yujia Li, Kevin Swersky, and Richard S. Zemel. Generative moment matching networks. *CoRR*, abs/1502.02761, 2015.
- [80] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. Advances in Neural Information Processing Systems, 33:6840–6851, 2020.
- [81] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. Advances in Neural Information Processing Systems, 34:8780–8794, 2021.
- [82] Calvin Luo. Understanding diffusion models: A unified perspective. *arXiv preprint arXiv:2208.11970*, 2022.
- [83] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. arXiv preprint arXiv:2202.00512, 2022.
- [84] Fred C Billingsley. Processing ranger and mariner photography. *Optical Engineering*, 4(4):404147, 1966.
- [85] Kavita Bala, Bruce Walter, and Donald P Greenberg. Combining edges and points for interactive high-quality rendering. ACM Transactions on Graphics (TOG), 22(3):631–640, 2003.
- [86] Jack Tumblin and Prasun Choudhury. Bixels: Picture samples with sharp embedded boundaries. In *Rendering Techniques*, pages 255–264. Citeseer, 2004.
- [87] Ganesh Ramanarayanan, Kavita Bala, and Bruce Walter. Feature-based textures, 2004.
- [88] Samuli Laine and Tero Karras. Efficient sparse voxel octrees. *IEEE Transactions on Visualization and Computer Graphics*, 17(8):1048–1059, 2010.
- [89] Evgueni Parilov and Denis Zorin. Real-time rendering of textures with feature curves. *ACM Transactions on Graphics (TOG)*, 27(1):1–15, 2008.
- [90] Darko Pavić and Leif Kobbelt. Two-colored pixels. In *Computer Graphics Forum*, volume 29, pages 743–752. Wiley Online Library, 2010.
- [91] Marco Agus, Enrico Gobbetti, José Antonio Iglesias Guitián, and Fabio Marton. Split-voxel: A simple discontinuity-preserving voxel representation for volume rendering. In VG@ Eurographics, pages 21–28, 2010.
- [92] Marco Tarini and Paolo Cignoni. Pinchmaps: Textures with customizable discontinuities. In *Computer Graphics Forum*, volume 24, pages 557–568. Blackwell Publishing, Inc Oxford, UK and Boston, USA, 2005.

- [93] Jörn Loviscach. Efficient magnification of bi-level textures. In ACM SIGGRAPH 2005 Sketches, pages 131–es. 2005.
- [94] Pradeep Sen, Mike Cammarano, and Pat Hanrahan. Shadow silhouette maps. ACM Transactions on Graphics (TOG), 22(3):521–526, 2003.
- [95] Pradeep Sen. Silhouette maps for improved texture magnification. In Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware, pages 65–73, 2004.
- [96] Shubham Tulsiani, Tinghui Zhou, Alexei A Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2626–2634, 2017.
- [97] Philipp Henzler, Niloy J Mitra, and Tobias Ritschel. Escaping plato's cave: 3d shape from adversarial rendering. In *ICCV*, pages 9984–9993, 2019.
- [98] Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. Hologan: Unsupervised learning of 3d representations from natural images. In *ICCV*, pages 7588–7597, 2019.
- [99] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhöfer. Deepvoxels: Learning persistent 3d feature embeddings, 2019.
- [100] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *IEEE CVPR*, pages 1125–1134, 2017.
- [101] Phong Nguyen, Animesh Karnewar, Lam Huynh, Esa Rahtu, Jiri Matas, and Janne Heikkila. Rgbd-net: Predicting color and depth images for novel views synthesis. arXiv preprint arXiv:2011.14398, 2020.
- [102] Kara-Ali Aliev, Artem Sevastopolsky, Maria Kolos, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics, 2020.

- [103] 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 CVPR*, pages 8798–8807, 2018.
- [104] Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. ACM Trans. Graph., 38(4):1– 14, 2019.
- [105] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. In ACM Trans. Graph., 2018.
- [106] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020.
- [107] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for antialiasing neural radiance fields. arXiv preprint arXiv:2103.13415, 2021.
- [108] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference* on Computer Vision and Pattern Recognition, pages 4578–4587, 2021.
- [109] 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. *arXiv preprint arXiv:2103.15595*, 2021.
- [110] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. arXiv preprint arXiv:2103.10380, 2021.
- [111] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. arXiv preprint arXiv:2103.14024, 2021.

- [112] Peter Hedman, Pratul P Srinivasan, Ben Mildenhall, Jonathan T Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. *arXiv preprint arXiv:2103.14645*, 2021.
- [113] 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.
- [114] Pengsheng Guo, Miguel Angel Bautista, Alex Colburn, Liang Yang, Daniel Ulbricht, Joshua M. Susskind, and Qi Shan. Fast and explicit neural view synthesis, 2021.
- [115] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. Atlasnet: A papier-mâché approach to learning 3d surface generation. *CoRR*, abs/1802.05384, 2018.
- [116] Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pigan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *IEEE CVPR*, pages 5799–5809, 2021.
- [117] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. *arXiv preprint arXiv:2007.02442*, 2020.
- [118] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *IEEE CVPR*, pages 11453–11464, 2021.
- [119] Thu Nguyen-Phuoc, Christian Richardt, Long Mai, Yong-Liang Yang, and Niloy J. Mitra. Blockgan: Learning 3d object-aware scene representations from unlabelled images. *CoRR*, abs/2002.08988, 2020.
- [120] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *NeurIPS*, 2020.
- [121] Julien N. P. Martel, David B. Lindell, Connor Z. Lin, Eric R. Chan, Marco Monteiro, and Gordon Wetzstein. Acorn: Adaptive coordinate networks for neural scene representation, 2021.

- [122] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, and Thomas Funkhouser. Local implicit grid representations for 3d scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.
- [123] Alex Yu, Sara Fridovich-Keil, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks, 2021.
- [124] Xiaodong Gu, Zhiwen Fan, Siyu Zhu, Zuozhuo Dai, Feitong Tan, and Ping Tan. Cascade cost volume for high-resolution multi-view stereo and stereo matching. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2495–2504, 2020.
- [125] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. *Advances in neural information processing systems*, 30, 2017.
- [126] Phong Nguyen, Animesh Karnewar, Lam Huynh, Esa Rahtu, Jiri Matas, and Janne Heikkila. Rgbd-net: Predicting color and depth images for novel views synthesis. In 2021 International Conference on 3D Vision (3DV), pages 1095–1105. IEEE, 2021.
- [127] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv* preprint arXiv:2304.07193, 2023.
- [128] Anchit Gupta, Wenhan Xiong, Yixin Nie, Ian Jones, and Barlas Oğuz. 3dgen: Triplane latent diffusion for textured mesh generation. *arXiv preprint arXiv:2303.05371*, 2023.
- [129] Sam Bond-Taylor and Chris G Willcocks. Gradient origin networks. arXiv preprint arXiv:2007.02798, 2020.
- [130] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

- [131] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *IEEE CVPR*, pages 4401–4410, 2019.
- [132] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *IEEE CVPR*, pages 8110–8119, 2020.
- [133] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. arXiv preprint arXiv:1809.11096, 2018.
- [134] Animesh Karnewar and Oliver Wang. Msg-gan: Multi-scale gradients for generative adversarial networks. In *IEEE CVPR*, pages 7799–7808, 2020.
- [135] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *arXiv preprint arXiv:2006.06676*, 2020.
- [136] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. arXiv preprint arXiv:2106.12423, 2021.
- [137] Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. arXiv preprint arXiv:2007.03898, 2020.
- [138] Alexander Alemi, Ben Poole, Ian Fischer, Joshua Dillon, Rif A Saurous, and Kevin Murphy. Fixing a broken elbo. In *International Conference on Machine Learning*, pages 159–168. PMLR, 2018.
- [139] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114, 2013.
- [140] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

- [141] Diederik P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible1x1 convolutions. *arXiv preprint arXiv:1807.03039*, 2018.
- [142] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. arXiv preprint arXiv:1605.08803, 2016.
- [143] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. arXiv preprint arXiv:1410.8516, 2014.
- [144] Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models. *arXiv preprint arXiv:2101.09258*, 2021.
- [145] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. arXiv preprint arXiv:2006.09011, 2020.
- [146] Alexia Jolicoeur-Martineau, Rémi Piché-Taillefer, Rémi Tachet des Combes, and Ioannis Mitliagkas. Adversarial score matching and improved sampling for image generation. arXiv preprint arXiv:2009.05475, 2020.
- [147] Jyoti Aneja, Alexander Schwing, Jan Kautz, and Arash Vahdat. Ncp-vae: Variational autoencoders with noise contrastive priors. *arXiv preprint arXiv:2010.02917*, 2020.
- [148] Stanislav Pidhorskyi, Donald A Adjeroh, and Gianfranco Doretto. Adversarial latent autoencoders. In *IEEE CVPR*, pages 14104–14113, 2020.
- [149] Aditya Grover, Manik Dhar, and Stefano Ermon. Flow-gan: Combining maximum likelihood and adversarial learning in generative models. In *AAAI*, 2018.
- [150] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. volume 48, pages 1558–1566, New York, New York, USA, 20–22 Jun 2016.
- [151] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. Singan: Learning a generative model from a single natural image. In *ICCV*, pages 4570–4580, 2019.

- [152] Tobias Hinz, Matthew Fisher, Oliver Wang, and Stefan Wermter. Improved techniques for training single-image gans. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1300–1309, 2021.
- [153] Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T Freeman, and Joshua B Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generativeadversarial modeling. In *Adv. Neural Inform. Process. Syst.*, pages 82–90, 2016.
- [154] Charlie Nash, Yaroslav Ganin, SM Ali Eslami, and Peter Battaglia. Polygen: An autoregressive generative model of 3d meshes. In *International Conference on Machine Learning*, pages 7220–7229. PMLR, 2020.
- [155] Giorgos Bouritsas, Sergiy Bokhnyak, Stylianos Ploumpis, Michael Bronstein, and Stefanos Zafeiriou. Neural 3d morphable models: Spiral convolutional networks for 3d shape representation learning and generation. In *ICCV*, pages 7213–7222, 2019.
- [156] Heli Ben-Hamu, Haggai Maron, Itay Kezurer, Gal Avineri, and Yaron Lipman. Multi-chart generative surface modeling. ACM Trans. Graph., 37(6):1–15, 2018.
- [157] Luca Cosmo, Antonio Norelli, Oshri Halimi, Ron Kimmel, and Emanuele Rodolà. Limp: Learning latent shape representations with metric preservation priors. In ECCV, pages 19–35. Springer, 2020.
- [158] Matheus Gadelha, Giorgio Gori, Duygu Ceylan, Radomir Mech, Nathan Carr, Tamy Boubekeur, Rui Wang, and Subhransu Maji. Learning generative models of shape handles. In *IEEE CVPR*, pages 402–411, 2020.
- [159] Yu Guo, Cameron Smith, Miloš Hašan, Kalyan Sunkavalli, and Shuang Zhao. Materialgan: reflectance capture using a generative svbrdf model. arXiv preprint arXiv:2010.00114, 2020.
- [160] Yuxuan Zhang, Wenzheng Chen, Huan Ling, Jun Gao, Yinan Zhang, Antonio Torralba, and Sanja Fidler. Image gans meet differentiable rendering for inverse graphics and interpretable 3d neural rendering. arXiv preprint arXiv:2010.09125, 2020.

- [161] Seung Wook Kim, Jonah Philion, Antonio Torralba, and Sanja Fidler. Drivegan: Towards a controllable high-quality neural simulation. In *IEEE CVPR*, pages 5820– 5829, 2021.
- [162] Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. Neural scene graphs for dynamic scenes. In *IEEE CVPR*, pages 2856–2865, June 2021.
- [163] Amlan Kar, Aayush Prakash, Ming-Yu Liu, Eric Cameracci, Justin Yuan, Matt Rusiniak, David Acuna, Antonio Torralba, and Sanja Fidler. Meta-sim: Learning to generate synthetic datasets. In *ICCV*, pages 4551–4560, 2019.
- [164] Terrance DeVries, Miguel Angel Bautista, Nitish Srivastava, Graham W. Taylor, and Joshua M. Susskind. Unconstrained scene generation with locally conditioned radiance fields. 2021.
- [165] Peng Zhou, Lingxi Xie, Bingbing Ni, and Qi Tian. Cips-3d: A 3d-aware generator of gans based on conditionally-independent pixel synthesis. arXiv preprint arXiv:2110.09788, 2021.
- [166] Jiatao Gu, Lingjie Liu, Peng Wang, and Christian Theobalt. Stylenerf: A stylebased 3d-aware generator for high-resolution image synthesis. arXiv preprint arXiv:2110.08985, 2021.
- [167] Jing Wen, Bi-Yi Chen, Chang-Dong Wang, and Zhihong Tian. Prgan: personalized recommendation with conditional generative adversarial networks. In 2021 IEEE International Conference on Data Mining (ICDM), pages 729–738. IEEE, 2021.
- [168] Katja Schwarz, Axel Sauer, Michael Niemeyer, Yiyi Liao, and Andreas Geiger. Voxgraf: Fast 3d-aware image synthesis with sparse voxel grids. Advances in Neural Information Processing Systems, 35:33999–34011, 2022.
- [169] 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.

- [170] Miguel Angel Bautista, Pengsheng Guo, Samira Abnar, Walter Talbott, Alexander Toshev, Zhuoyuan Chen, Laurent Dinh, Shuangfei Zhai, Hanlin Goh, Daniel Ulbricht, et al. Gaudi: A neural architect for immersive 3d scene generation. Advances in Neural Information Processing Systems, 35:25102–25116, 2022.
- [171] Ivan Skorokhodov, Sergey Tulyakov, Yiqun Wang, and Peter Wonka. Epigraf: Rethinking training of 3d gans. Advances in Neural Information Processing Systems, 35:24487–24501, 2022.
- [172] Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojcic, and Sanja Fidler. Get3d: A generative model of high quality 3d textured shapes learned from images. *Advances In Neural Information Processing Systems*, 35:31841–31854, 2022.
- [173] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. *Advances in Neural Information Processing Systems*, 34:6087–6101, 2021.
- [174] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 2837–2845, 2021.
- [175] Linqi Zhou, Yilun Du, and Jiajun Wu. 3d shape generation and completion through point-voxel diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5826–5835, 2021.
- [176] Xiaohui Zeng, Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, and Karsten Kreis. Lion: Latent point diffusion models for 3d shape generation. arXiv preprint arXiv:2210.06978, 2022.
- [177] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim

Bibliography

Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022.

- [178] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: Highresolution text-to-3d content creation. In *Proceedings of the IEEE/CVF Conference* on Computer Vision and Pattern Recognition, pages 300–309, 2023.
- [179] Daniel Watson, William Chan, Ricardo Martin-Brualla, Jonathan Ho, Andrea Tagliasacchi, and Mohammad Norouzi. Novel view synthesis with diffusion models. arXiv preprint arXiv:2210.04628, 2022.
- [180] Jiatao Gu, Alex Trevithick, Kai-En Lin, Joshua M Susskind, Christian Theobalt, Lingjie Liu, and Ravi Ramamoorthi. Nerfdiff: Single-image view synthesis with nerf-guided distillation from 3d-aware diffusion. In *International Conference on Machine Learning*, pages 11808–11826. PMLR, 2023.
- [181] Luke Melas-Kyriazi, Iro Laina, Christian Rupprecht, and Andrea Vedaldi. Realfusion: 360deg reconstruction of any object from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8446– 8455, 2023.
- [182] Zhizhuo Zhou and Shubham Tulsiani. Sparsefusion: Distilling view-conditioned diffusion for 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12588–12597, 2023.
- [183] Congyue Deng, Chiyu Jiang, Charles R Qi, Xinchen Yan, Yin Zhou, Leonidas Guibas, Dragomir Anguelov, et al. Nerdi: Single-view nerf synthesis with language-guided diffusion as general image priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20637–20647, 2023.
- [184] Tengfei Wang, Bo Zhang, Ting Zhang, Shuyang Gu, Jianmin Bao, Tadas Baltrusaitis, Jingjing Shen, Dong Chen, Fang Wen, Qifeng Chen, et al. Rodin: A generative model

Bibliography

for sculpting 3d digital avatars using diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4563–4573, 2023.

- [185] Norman Müller, Yawar Siddiqui, Lorenzo Porzi, Samuel Rota Bulo, Peter Kontschieder, and Matthias Nießner. Diffrf: Rendering-guided 3d radiance field diffusion. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 4328–4338, 2023.
- [186] Yinbo Chen, Sifei Liu, and Xiaolong Wang. Learning continuous image representation with local implicit image function. *CoRR*, abs/2012.09161, 2020.
- [187] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 5939–5948, 2019.
- [188] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision* and Pattern Recognition, pages 165–174, 2019.
- [189] 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.
- [190] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.
- [191] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*, 2018.
- [192] Lin Yen-Chen. Nerf-pytorch. https://github.com/yenchenlin/ nerf-pytorch/, 2020.

- [193] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings* of the IEEE conference on computer vision and pattern recognition, pages 586–595, 2018.
- [194] 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*, pages 441–459. Springer, 2020.
- [195] Tong Wu, Jiarui Zhang, Xiao Fu, Yuxin Wang, Jiawei Ren, Liang Pan, Wayne Wu, Lei Yang, Jiaqi Wang, Chen Qian, et al. Omniobject3d: Large-vocabulary 3d object dataset for realistic perception, reconstruction and generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 803–814, 2023.
- [196] Rameen Abdal, Peihao Zhu, Niloy J Mitra, and Peter Wonka. Styleflow: Attributeconditioned exploration of stylegan-generated images using conditional continuous normalizing flows. ACM Trans. Graph., 40(3):1–21, 2021.
- [197] Yujun Shen, Ceyuan Yang, Xiaoou Tang, and Bolei Zhou. Interfacegan: Interpreting the disentangled face representation learned by gans. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2020.
- [198] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. Ganspace: Discovering interpretable gan controls. *arXiv preprint arXiv:2004.02546*, 2020.
- [199] Thu Nguyen-Phuoc, Christian Richardt, Long Mai, Yong-Liang Yang, and Niloy Mitra. Blockgan: Learning 3d object-aware scene representations from unlabelled images. arXiv preprint arXiv:2002.08988, 2020.
- [200] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.
- [201] Philipp Henzler, Niloy J Mitra, , and Tobias Ritschel. Learning a neural 3d texture space from 2d exemplars. In *CVPR*, June 2019.

- [202] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pages 8162– 8171. PMLR, 2021.
- [203] Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space. *Advances in neural information processing systems*, 34:11287–11302, 2021.
- [204] Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In ACM SIGGRAPH 2022 Conference Proceedings, pages 1–10, 2022.
- [205] Chenlin Meng, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Image synthesis and editing with stochastic differential equations. arXiv preprint arXiv:2108.01073, 2021.
- [206] Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. J. Mach. Learn. Res., 23(47):1–33, 2022.
- [207] Ka-Hei Hui, Ruihui Li, Jingyu Hu, and Chi-Wing Fu. Neural wavelet-domain diffusion for 3d shape generation. In SIGGRAPH Asia 2022 Conference Papers, pages 1–9, 2022.
- [208] Philipp Henzler, Jeremy Reizenstein, Patrick Labatut, Roman Shapovalov, Tobias Ritschel, Andrea Vedaldi, and David Novotny. Unsupervised learning of 3d object categories from videos in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4700–4709, 2021.
- [209] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision*, pages 333–350. Springer, 2022.

- [210] 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. Advances in neural information processing systems, 30, 2017.
- [211] Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying mmd gans. arXiv preprint arXiv:1801.01401, 2018.
- [212] Michael Niemeyer and Andreas Geiger. Campari: Camera-aware decomposed generative neural radiance fields. In 2021 International Conference on 3D Vision (3DV), pages 951–961. IEEE, 2021.
- [213] Gowthami Somepalli, Vasu Singla, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Diffusion art or digital forgery? investigating data replication in diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6048–6058, 2023.
- [214] Titas Anciukevičius, Zexiang Xu, Matthew Fisher, Paul Henderson, Hakan Bilen, Niloy J Mitra, and Paul Guerrero. Renderdiffusion: Image diffusion for 3d reconstruction, inpainting and generation. In *Proceedings of the IEEE/CVF Conference* on Computer Vision and Pattern Recognition, pages 12608–12618, 2023.
- [215] Zifan Shi, Sida Peng, Yinghao Xu, Andreas Geiger, Yiyi Liao, and Yujun Shen. Deep generative models on 3d representations: A survey. *arXiv preprint arXiv:2210.15663*, 2022.
- [216] Kyle Gao, Yina Gao, Hongjie He, Dening Lu, Linlin Xu, and Jonathan Li. Nerf: Neural radiance field in 3d vision, a comprehensive review. arXiv preprint arXiv:2210.00379, 2022.
- [217] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser.
 Ibrnet: Learning multi-view image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2021.

- [218] Xiaoxiao Long, Cheng Lin, Peng Wang, Taku Komura, and Wenping Wang. Sparseneus: Fast generalizable neural surface reconstruction from sparse views. In *European Conference on Computer Vision*, pages 210–227. Springer, 2022.
- [219] Xingkui Wei, Yinda Zhang, Zhuwen Li, Yanwei Fu, and Xiangyang Xue. Deepsfm: Structure from motion via deep bundle adjustment. In *Computer Vision–ECCV 2020:* 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16, pages 230–247. Springer, 2020.
- [220] Xingkui Wei, Yinda Zhang, Xinlin Ren, Zhuwen Li, Yanwei Fu, and Xiangyang Xue. Deepsfm: Robust deep iterative refinement for structure from motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.
- [221] Aji Resindra Widya, Akihiko Torii, and Masatoshi Okutomi. Structure-from-motion using dense cnn features with keypoint relocalization. arXiv e-prints, pages arXiv– 1805, 2018.
- [222] 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.
- [223] Eric R. Chan, Koki Nagano, Matthew A. Chan, Alexander W. Bergman, Jeong Joon Park, Axel Levy, Miika Aittala, Shalini De Mello, Tero Karras, and Gordon Wetzstein. GeNVS: Generative novel view synthesis with 3D-aware diffusion models. In *arXiv*, 2023.
- [224] Minghua Liu, Chao Xu, Haian Jin, Linghao Chen, Mukund Varma T, Zexiang Xu, and Hao Su. One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization. *Advances in Neural Information Processing Systems*, 36, 2024.
- [225] Yuan Liu, Sida Peng, Lingjie Liu, Qianqian Wang, Peng Wang, Christian Theobalt, Xiaowei Zhou, and Wenping Wang. Neural rays for occlusion-aware image-based

Bibliography

rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7824–7833, 2022.

- [226] Towaki Takikawa, Alex Evans, Jonathan Tremblay, Thomas Müller, Morgan McGuire, Alec Jacobson, and Sanja Fidler. Variable bitrate neural fields. In ACM SIGGRAPH 2022 Conference Proceedings, pages 1–9, 2022.
- [227] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis.
 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023.
- [228] Reiner Birkl, Diana Wofk, and Matthias Müller. Midas v3. 1–a model zoo for robust monocular relative depth estimation. *arXiv preprint arXiv:2307.14460*, 2023.
- [229] Shariq Farooq Bhat, Reiner Birkl, Diana Wofk, Peter Wonka, and Matthias Müller. Zoedepth: Zero-shot transfer by combining relative and metric depth. *arXiv preprint arXiv:2302.12288*, 2023.
- [230] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. *arXiv* preprint arXiv:2401.10891, 2024.
- [231] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. Synsin: End-to-end view synthesis from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7467–7477, 2020.
- [232] Ruslan Rakhimov, Andrei-Timotei Ardelean, Victor Lempitsky, and Evgeny Burnaev. Npbg++: Accelerating neural point-based graphics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15969–15979, 2022.
- [233] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

- [234] Tong Wu, Jiarui Zhang, Xiao Fu, Yuxin Wang, Jiawei Ren, Liang Pan, Wayne Wu, Lei Yang, Jiaqi Wang, Chen Qian, et al. Omniobject3d github code and dataset, 2023. https://github.com/omniobject3d/OmniObject3D.
- [235] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals*, *Systems & Computers*, 2003, volume 2, pages 1398–1402. Ieee, 2003.
- [236] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.
- [237] Ziang Cao, Fangzhou Hong, Tong Wu, Liang Pan, and Ziwei Liu. Large-vocabulary3d diffusion model with transformer. *arXiv preprint arXiv:2309.07920*, 2023.
- [238] William Peebles and Saining Xie. Scalable diffusion models with transformers. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 4195–4205, 2023.
- [239] Jiawei Yang, Katie Z Luo, Jiefeng Li, Kilian Q Weinberger, Yonglong Tian, and Yue Wang. Denoising vision transformers. arXiv preprint arXiv:2401.02957, 2024.
- [240] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. arXiv preprint arXiv:2307.05663, 2023.
- [241] Tianchang Shen, Jacob Munkberg, Jon Hasselgren, Kangxue Yin, Zian Wang, Wenzheng Chen, Zan Gojcic, Sanja Fidler, Nicholas Sharp, and Jun Gao. Flexible isosurface extraction for gradient-based mesh optimization. ACM Transactions on Graphics (TOG), 42(4):1–16, 2023.

List of Figures

1.1	(a) A screenshot from the highest grossing game of 2023 titled "God	
	of War: Ragnarok", and (b) A frame from the 2022 blockbuster	
	movie titled "Avatar: The way of water"	12
1.2	Samples generated by Stable-Diffusion [12] for the prompts (a) "A	
	large public music concert on Mars"; (b) "A Unicorn swimming in	
	the ocean"; (c) "Neanderthals having a candle light dinner"	15
1.3	Examples of live imaginative painting from the demo created by	
	the X-user '@MartinNebelong'. The user is drawing an artistic tree	
	branch in (a), while drawing a realistic portrait in (b). The ControlNet	
	transforms the user's vector input into the specified styles of RGB	
	<pre>images in real-time. Ref: https://x.com/AnimeshKarnewar/</pre>	
	status/1759997147133444194	16
1.4	Examples of textures generated by the Text2Tex method from Chen	
	et al. [20] given pre-made 3D meshes and user provided text-prompts.	17
1.5	Figure shows 2D orthographic projection of three 3D representations	
	namely point-cloud (left), mesh (middle) and voxel-grid (right)	21
2.1	A Venn diagram showing where Deep Learning fits in the hierarchy	
	of the concepts in AI.	24
2.2	The figure describes the flow of information in a typical Ma-	
	chine Learning framework. Both the example_input and	
	example_output are the same for self-supervised learning, while	
	in the case of unsupervised learning (K-means for instance), the	
	objective function doesn't take as input any example_output	25

List of Figures

2.3	The perceptron algorithm (left) was the basis of the modern-day	
	neural-network models used today. (Right) shows how the individual	
	perceptrons are arranged in case of a Multi-Layered Perceptrons model.	26
2.4	Differentiable volumetric rendering of a continuous volumetric 3D	
	radiance field denoted by \mathcal{R}	34
2.5	Two perceptually very similar images \mathcal{I}_1 and \mathcal{I}_2 have the maximum	
	possible L2 distance in the 4×4 image space where each pixel can	
	have values strictly between $[0, 1]$, s.t. 0 represents the colour black	
	and 1 represents the colour white.	39
2.6	The interaction between the two networks, viz. Generator $G(z)$ and	
	the Discriminator $D(x)$ (for real sample input) or $D(\hat{x})$ (for fake	
	sample input) during mini-max game of the GAN training	39
2.7	The forward and backward markov chains defined by Diffusion	
	Models. The first step x_0 denotes the true data samples such as	
	images while the final step x_T indicates pure Gaussian noise. All the	
	intermediate x_t represent noisy versions of the data-samples	41
4.1	We present a method to represent complex signals such as images	
	or 3D scenes, both volumetric (left) and surface (right), on regularly	
	sampled grid vertices. Our method is able to match the expres-	
	siveness of coordinate-based MLPs while retaining reconstruction	
	and rendering speed of voxel grids, without requiring any neural	
	networks or sparse data structures	53
4.2	Representing a ground-truth function (blue) in a 1D (a) and 2D	
	(b) grid cell using the linear basis (yellow) and a ReLU-Fields	
	(pink). The reference has a c1-discontinuity inside the domain that a	
	linear basis cannot capture. A ReLU-Field will pick two values y_1	
	and y_2 , such that their interpolation, after clamping will match the	
	sharp c1-discontinuity in the ground-truth (blue) function	55

4.3	Representing an image with a standard pixel grid bi-linearly inter-	
	polated to a larger size (Grid) versus a ReLU-Field of the same size	
	(ReLUField). The grid-size of the variants, ReLUField and Grid, is	
	64x smaller; while of, ReLUFieldL and GridL, is 32x smaller than	
	the source image-resolution along each dimension. Note that the 'L'	
	variants have a bigger grid-size and hence less smaller than the GT	
	raster image. Simply adding a ReLU allows for significantly more	
	sharpness and detail to be expressed. Hence, we can say that the	
	humble ReLU is truly the little non-linearity that could	57
4.4	Qualitative comparison between NeRF-PT, GridL and ReLUFieldL.	
	Grid-based versions converge much faster, and we can see significant	
	sharpness improvements of ReLUFieldL over GridL, for example in	
	the leaves of the plant. See also supplementary video.	58
4.5	Qualitative results for the real-captured scene extension of ReLU-	
	Fields on Flowers. We decompose the scene into a series of spherical-	
	background shells and a foreground ReLU-Field layer, which are	
	alpha-composited together to give final novel view renderings. The	
	top-left visualization shows the composite of the background spheri-	
	cal shells un-projected onto a 2D image-plane	62
4.6	Qualitative results for the occupancy fields comparing Grid, MLP,	
	and ReLUField.	64
5.1	Single scene 3D remixes. We introduce 3INGAN that takes a set	
	of 2D photos of a <i>single</i> self-similar scene to produce a generative	
	model of 3D scene remixes, each of which can be rendered from	
	arbitrary camera configurations, without any flickering or spatio-	
	temporal artifacts. Bottom row insets show zooms from different	
	generative samples, rendered from the same camera view, to high-	
	light the quality and diversity of the results.	68

5.2	3INGAN setup. Overview of our approach with two parts: an	
	initialization of a reference 3D feature grid (top) and a stage-wise	
	learning of a generative model (bottom). Input to the system is a set	
	of 2D images seen on the top left. From these, optimization using	
	differentiable rendering for known views produces the reference	
	feature grid, which is the input to the next step. The rows below	
	("Level") denote levels of training the generator, a 2D discriminator,	
	and a 3D discriminator. The 3D discriminator (right) gets random	
	3D patches from the reference or generated 3D grid, while the 2D	
	discriminator (right) gets random 2D patches from reference or from	
	generated renderings.	71
5.3	Datasets. Example renderings of the scenes from our synthetic and	
	real world datasets (Blocks, Chalk).	76
5.4	Single 3D scene FID. We extend FID scores to our single scene use	
	case. The distribution of feature responses is computed for different	
	camera views (rows) and generations (columns). The reference (left)	
	leads to a certain distribution of features. Rather than matching	
	the reference distribution across all views and all seeds (red lines),	
	we compare it to the distribution of a single fixed seed across all	
	views (green lines) to measure visual quality. Then, we compare the	
	variance of the distribution of features across all seeds under a fixed	
	view as a measure of <i>scene diversity</i>	78
5.5	Qualitative comparison. Comparison of visual quality for different	
	methods (columns) for different scenes (rows).	78
5.6	Diversity across different generative samples. Diversity under	
	changing seeds (columns) of different methods (rows) for different	
	scenes (left and right blocks). See also Figure 5.3	79
5.7	Scene retargetting. Retargeting the Plants, the Logs, and Fish	
	scenes to novel aspect ratios. Since ours is CNN-based, it is easy to	
	retarget scenes to different sizes.	80

6.1	The 3inFusion pipeline takes as input random 3D crops of the fitted	
	ReLU-Field grid. This input is then combined with time-conditional	
	Gaussian noise, which is then denoised by the 3D-Unet. During	
	inference, we use this 3D crop denoiser to denoise a noise grid of	
	the original grid-size iteratively to generate semantically meaningful	
	variations of the original scene	84
6.2	Given a ReLU-field grid of a 3D scene, 3inFusion can generate	
	semantically meaningful variations of it, much better than 3inGAN,	
	while training in a simple and stable manner	85
6.3	We present HoloDiffusion as the first 3D-aware generative diffu-	
	sion model that produces 3D-consistent images and is trained with	
	only posed image supervision. Here we show a few different sam-	
	ples generated from models trained on different classes of the CO3D	
	dataset [76]	86
6.4	Method overview. Our HoloDiffusion takes as input video frames	
	for category-specific videos $\{s^i\}$ and trains a diffusion-based gen-	
	erative model \mathcal{D}_{θ} . The model is trained with only posed image	
	supervision $\{(I_j^i, P_j^i)\}$, without access to 3D ground-truth. Once	
	trained, the model can generate view-consistent results from novel	
	camera locations.	87
6.5	View consistency. Evaluation of the consistency of the shape ren-	
	ders under camera motion. While our results (top) remain consistent,	
	pi-GAN [116]'s results (bottom) suffer from significant appearance	
	variations across view changes.	90
6.6	Comparisons. Samples generated by our HoloDiffusion compared	
	to those by pi-GAN, EG3D, and GET3D.	92
6.7	Sampling across time. Rendering of HoloDiffusion's iterative	
	sampling process for a hydrant and a teddy bear. The diffusion time	
	decreases from left ($t = T = 1000$) to the right ($t = 0$)	94

7.1	We propose HoloFusion to generate photo-realistic 3D radiance
	fields by extending the HoloDiffusion method with a jointly trained
	2D 'super resolution' network. The independently super-resolved im-
	ages are fused back into the 3D representation to improve the fidelity
	of the 3D model via distillation, while preserving the consistency
	across view changes
7.2	Overview. HoloFusion, which trains the 3D denoiser network D_{θ} ,
	is augmented with the 2D 'super-resolution' diffusion model D_{β} .
	Both models are trained end-to-end by supervising their outputs with
	2D photometric error
7.3	Distillation. HoloFusion distills a single high-resolution voxel grid
	V_0^H by minimizing a top-k patch-remix loss \mathcal{L}_{distil} between the grid
	renders $R_{\eta'}(V_0^H, C)$ and a bank \mathcal{I}_C of $K = 5$ high-res images output
	by the 2D diffusion upsampler D_{β} for each scene camera $C.$ 104
7.4	Generated 3D samples visualized from a moving camera. π -GAN
	and HoloDiffusion* fail to produce 3D view consistent samples,
	while DreamFusion suffers from the "Janus" problem (multiple
	heads)
7.5	Fusing views. Our patch-remix (section 7.3.2.2) compared to the
	SDS and MSE distillation. MSE has "floaters" and viewpoint in-
	consistencies, SDS over-smooths the texture. Ours is robust and
	produces superior quality
7.6	3D samples generated by our HoloFusion compared to π -GAN,
	EG3D, GET3D, HoloDiffusion, HoloDiffusion*, and the text-to-3D
	Stable-DreamFusion
7.7	Heatmaps illustrating the per-pixel color variance of $K = 10$ hypoth-
	esis produced by the upsampler D_{β} . Some samples contain artifacts
	around the object boundaries which correspond to the high-variance
	regions in the figure. Our top-K patch-remix increases robustness by
	allowing the loss to discard such artifacts during distillation 109

8.1	We propose the GOEmbed (Gradient Origin Embedding) mecha-
	nism that encodes source views (o^{ctxt}) and camera parameters (ϕ^{ctxt})
	into arbitrary 3D Radiance-Field representations $g(c,d)$ (sec. 8.3).
	We show how these general-purpose GOEmbeddings can be used in
	the context of 3D DFMs (Diffusion with Forward Models) (sec. 8.5)
	and for sparse-view 3D reconstruction (sec. 8.6)
8.2	GOEmbed illustration. We demonstrate the mechanism here using
	the Triplane representation for $g(c,d)$, but note that this can be
	applied to other representations as well. The GOEmbed mechanism
	(eq. 8.1) consists of two steps. First we render the origin ζ_0 from
	the context-poses ϕ^{ctxt} ; then we compute the gradient of the MSE
	between the renders and the source-views o^{ctxt} wrt. the origin ζ_0
	which gives us the GOEmbed encodings ζ_{enc}
8.3	Plenoptic Encoding Qualitative Evaluation. The rows MLP, Tri-
	plane and Voxel-grid show the renders of the GOEmbed encoded
	representations from the target-view respectively. The colour-coded
	columns demonstrate the effect of varying the number-of-source
	views (1, 2, 3, 4) used in the GOEmbed encoding. The SSO column
	shows the target render of the single-scene-overfitted representa-
	tion while the G.T. column shows the mesh-render from the dataset
	(repeated for clarity)
8.4	3D Generation Qualitative Evaluation. 3D samples generated by
	our GOEmbedFusion compared to the prior GAN, and Diffusion
	based baselines

List of Tables

4.1	Evaluation results on 3D synthetic scenes. Metrics used are PSNR	
	(†) / LPIPS (\downarrow). The column NeRF-TF* quotes PSNR values from	
	prior work [34], and as such we do not have a comparable runtime	
	for this method.	60
4.2	Evaluation results on modeling 3D geometries as occupancy fields.	
	Metric used is Volumetric-IoU [32]. The baseline MLP is our imple-	
	mentation of OccupancyNetworks [32]	63
5.1	Comparisons and ablations. We enumerate the different methods	
	based on how they make use of 2D versus 3D information, and if	
	they operate on a single scene or multiple scenes	76
5.2	Quality versus diversity. A good generative model should have	
	a good mix of quality and diversity - excellent quality with no	
	diversity or vice versa are both undesirable. Visual Quality and Scene	
	Diversity for different methods (columns) and different data sets	
	(rows). To simplify comparison, we normalize the numbers so that	
	ours is always 1. The best for each metric on each dataset is bolded	
	and second best is <u>underlined</u> . Please refer to the supplementary for	
	unscaled numbers.	79

6.1	Quantitative evaluation . FID and KID on 4 classes of CO3Dv2
0.1	comparing our HoloDiffusion with the baselines pi-GAN [116]
	EG3D [169] GET3D [172] and the non-bootstrapped version of
	our HoloDiffusion. The column "VP" denotes whether renders of a
	method are 3D view-consistent or not
7.1	FID (\downarrow) and KID (\downarrow) on 4 classes of Co3Dv2 [76]. We compare
	with 3D generative modeling baselines (rows 1-5); with an SDS
	distillation-based Stable-DreamFusion (row 6); and with ablations
	of our HoloFusion (rows 7–8). The column "VP" denotes whether
	renders of a method are 3D view-consistent or not
8.1	Plenoptic Encoding Quantitative Evaluation. $PSNR(\uparrow)$, $LPIPS(\downarrow)$
	and $SSIM(\uparrow)$ reported on three different representations of the 3D
	Radiance-Field g, namely, Triplanes, Voxel-Grids and MLPs. All
	the metrics are evaluated for target views (different from the source
	views) against the G.T. mesh renders from the dataset. The SSO
	(Single Scene Overfitting) scores denote the case of individually
	fitting the representations to the 3D scenes
8.2	3D Generation Quantitative Evaluation. $FID(\downarrow)$ and $KID(\downarrow)$
	scores on the OmniObject3D dataset comparing our GOEmbed-
	Fusion with GAN baselines EG3D [169], and GET3D[172]; with
	the non-forward diffusion baselines DiffRF[185], DiffTF[237], and
	Our non-forward diffusion baseline; and, with the DFM (Diffusion
	with Forward Model) [222]
8.3	3D reconstruction Quantitative Evaluation. $PSNR(\uparrow)$, $LPIPS(\downarrow)$
	and $SSIM(\downarrow)$ of our GOEmbed reconstruction model, and GOEm-
	bedFusion's "pseudo"-deterministic 3D reconstruction output com-
	pared to LRM baselines. We again include the SSO (Single Scene
	Overfitting) here for comparison

Abbreviations

- AI Artificial Intelligence 21
- AR Augmented Reality 9
- CGI Computer-Generated Imagery 8, 9, 19
- CNN Convolutional Neural Network 24, 25
- CS Computer Science 20
- GAN Generative Adversarial Network 35–37
- **GM** Generative Modelling 3, 10
- HCI Human Computer Interface 9
- LLM Large Language Model 3, 12
- ML Machine Learning 11, 21, 22, 28, 34, 37, 39
- MLP Multi Layer Perceptron 3, 24
- NN Neural Network 24–28, 30
- **RNN** Recurrent Neural Network 25
- SGD Stochastic Gradient Descent 26, 28
- VAE Variational Auto-Encoder 33
- VR Virtual Reality 9