Exploring the Capabilities of Automatic 3D Model Generation: A Comparative Study

Bachelor's Thesis

in the applied computer science course of the faculty of business informatics and applied computer science at the Otto-Friedrich-University of Bamberg

Faculty of Computer Graphics

Author: Andreas Franz Schwab

Examiner: Prof. Dr. Sophie Jörg

Abstract

The exploration of automatic 3D model generation represents a significant stride in the realm of computer vision and machine learning. This thesis delves into the capabilities of this innovative technology, focusing on a comparative study of various methodologies that facilitate the creation of 3D models from textual and image inputs. The importance of this topic stems from the increasing demand for efficient and accurate 3D model generation in various applications, ranging from virtual reality and gaming to medical imaging.

Despite the advancements in this field, there exists a research gap in comprehensively understanding and comparing different automatic 3D model generation techniques, particularly in the context of their effectiveness, accuracy, and applicability. This thesis aims to bridge this gap by conducting a thorough analysis of selected methods, including Dreamfusion, Fantasia3D, and Magic 3D for text input, and Magic 123 and Wonder3D for image input. The research methods include a detailed experimental setup that uses both subjective evaluations and performance metrics to analyze these technologies.

The key message of this thesis highlights the differentiated capabilities and limitations of the individual methods and provides insights into their applicability and potential for future development. The results show significant differences in the accuracy and efficiency of the methods examined and highlight the strengths and weaknesses of the individual techniques in different scenarios.

By offering a comprehensive comparison of various methodologies in automatic 3D model generation, this thesis not only aids in the understanding of these technologies but also paves the way for future research, particularly in addressing generative biases and exploring emerging trends.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In today's rapidly evolving digital landscape, the demand for 3D models continues to grow, driven by the ever-increasing need for immersive and realistic visual experiences. Generative AI techniques have proven to be a transformative force, enabling researchers and practitioners to develop innovative methods that automate the complicated process of creating 3D models. These remarkable innovations can reshape our digital interactions by enabling advanced simulations, in-depth analysis and captivating visualizations of complex real-world phenomena.

Starting out in 3D synthesis can be a challenging experience, particularly for novices with limited prior knowledge. While directly applying the models outlined in Chapter 3 may appear straightforward, acquiring a more in-depth understanding of the diverse methodologies and their foundational principles greatly enriches the learning process. This deeper understanding not only improves the practical application of these models, but also enables future advances in the field of automatic 3D model generation.

This thesis provides a comprehensive examination into various models and technologies in automated 3D model generation, offering a detailed analysis of their mechanisms, capabilities, and limitations. The study focuses on evaluating the technologies' effectiveness, emphasizing their proficiency in creating functionally robust and aesthetically appealing 3D models. This research contributes to the fields of computer graphics and artificial intelligence, serving as a valuable resource for novices in automatic 3D model generation and inspiring future researchers.

To provide a foundation for this exploration, a comprehensive examination of the fundamentals of 2D generative AI is undertaken. This includes a comprehensive understanding of key generative models such as Variational Autoencoders (VAEs) [Kingma and Welling, 2013; Rezende et al., 2014], Generative Adversarial Networks (GANs) [Goodfellow et al., 2020] and Diffusion Models [Yang et al., 2022; Ho et al., 2020; Sohl-Dickstein et al., 2015]. Additionally, a brief introduction to Contrastive Language-Image Pre-training (CLIP) [Radford et al., 2021], Stable Diffusion [Rombach et al., 2022] and Multilayer Perceptron (MLP) is given. The section concludes with an overview of various forms of 3D representation, including Meshes, Point-clouds, Voxels, Neural Radiance Fields (NeRFs) [Mildenhall et al., 2021], Deep Marching Tetrahedra (DMTets) [Shen et al., 2021], and Instant Neural Graphics Primitives (InstantNGPs) [Müller et al., 2022].

This research further evaluates various approaches to generating 3D models based on

both images and text input. The methods examined include DreamFusion [Poole et al., 2022], Magic3D [Lin et al., 2023], Fantasia3D [Chen et al., 2023], Magic123 [Qian et al., 2023] and Wonder3D [Long et al., 2023]. Each method presents unique challenges and opportunities, and their results are thoroughly examined through a comparative study.

The evaluation involves the application of rigorous performance evaluation metrics covering a spectrum of critical aspects. These metrics include the accuracy of the input and results to ensure that the generated models accurately match the input and maintain the intended properties. Also, the level of detail of these models is examined to assess their ability to capture complicated features and nuances. An important criterion is realism, i.e. the ability of the generated models to reflect the authenticity of their real-life counterparts. Furthermore, technical aspects such as symmetry and model integrity are also examined to check the structural soundness and coherence of the generated 3D models. This comprehensive analysis leads to an in-depth understanding of the strengths and weaknesses of the individual methods investigated.

In addition, this thesis casts a discerning eye towards the evolving landscape of 3D modeling and highlights emerging trends that have the potential to reshape the field. These trends include innovations such as video-to-3D methods that open innovative dimensions in the creation of three-dimensional scenes. Also, the important topic of inherited biases is discussed, highlighting the need for more in-depth research to ensure the fairness of these methods. By exploring these unexplored areas, the aim is to promote a fairer and more progressive future for the field of 3D modeling.

Chapter 2

Basics

This chapter provides the groundwork necessary for the comparative analysis of automatic 3D model generation techniques by introducing the key technologies that drive these methods. It is essential to have a common understanding of the 2D generative models and 3D data representations that form the basis of this field.

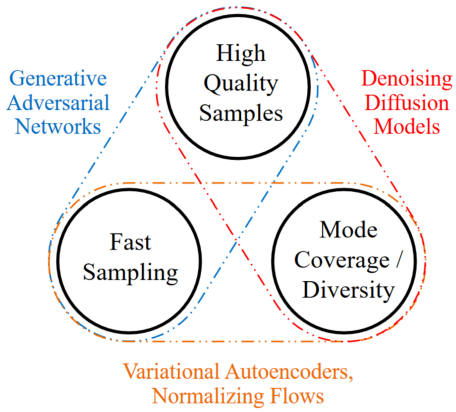


Figure 1: The Generative Learning Trilemma: Balancing Quality, Speed, and Diversity in Generative Models. Image taken from [Xiao et al., 2021].

The chapter starts with a brief overview of the most common generative text-to-2D models, as they play a central role for 3D synthesis. The section aims to shed light on the "generative learning trilemma", a concept introduced in Figure 1 by Xiao et al. that describes the balance between quality, speed and diversity in different generative models. The chapter includes Variational Autoencoders (VAEs) [Kingma and Welling, 2013; Rezende et al., 2014], which provide a basic probabilistic framework for learning complex data representations while balancing fast sampling and diversity in their results. Following this, Generative Adversarial Networks (GANs) [Goodfellow et al., 2020] are presented, highlighting their fast training mechanisms for producing high-quality samples that include both generator and discriminator parts. In addition, the chapter explores the area of diffusion models, which are characterized by diversity and sample quality. A particular focus is on Denoising Diffusion Probabilistic Models (DDPMs) [Ho et al., 2020; Sohl-Dickstein et al., 2015], while Score-Based generative Models (SGMs) [Song and

Ermon, 2019] and Stochastic Differential Equations (SDEs) [Song et al., 2020, 2021] are also addressed.

Furthermore, the chapter deals with Contrastive Language-Image Pre-training (CLIP) [Radford et al., 2021] and illustrates its effectiveness in combining natural language and visual data, which is crucial for text-driven 3D model generation. Stable Diffusion [Rombach et al., 2022] is then briefly introduced which incorporates the before explained Diffusion Models, VAEs and CLIP to generate images from text prompts. Additionally, the architecture of basic Multilayer Perceptrons (MLPs) are briefly described. The final part of the chapter is dedicated to exploring different forms of 3D data representation, such as meshes, point clouds, and voxels, as well as some more efficient methods for representing 3D scenes, e.g., NeRFs, DMTets, and InstantNGP. Understanding these representations is key to understanding the structural makeup of 3D objects in computational environments.

2.1 Variational Autoencoders – VAEs

Generative models play a pivotal role in the field of machine learning, particularly in tasks involving the synthesis and understanding of complex data distributions. Among the diverse types of generative models, Variational Autoencoders (VAEs) stand out for their unique approach and application. These models have gained prominence for their ability to efficiently generate new data samples while capturing the structures of complex data distributions. VAEs operate on the principle of encoding input data into a latent space, a compressed representation of the input data containing only the most imporant features, and then reconstructing the input from this space. This process is governed by an explicit probabilistic framework, where the model is trained to maximize the likelihood of the data under the learned distribution. This explicit modeling of data distribution in the latent space not only help the generation of new data but also provides valuable insights into the data's inherent characteristics.

VAEs are essentially based on the architecture of Autoencoders, which apply an iterative process to identify the optimal encoder and decoder pair, aiming to minimize the reconstruction loss while preserving important information after dimensionality reduction. The encoder compresses the input data into a low-dimensional representation, or "code" [Hinton and Salakhutdinov, 2006; Goodfellow et al., 2016], which captures the most relevant features of the input, within a latent space. The decoder's role is to reconstruct the original input from this compressed latent representation. The reconstruction error, which is the discrepancy between the output and the original data, is then used to backpropagate and optimize the model's weights. This process, as described in works by Hinton and Salakhutdinov, Goodfellow et al., and Michelucci, strikes a balance between data compression and information preservation.

Autoencoders, by design, focus on approximate replication of input data rather than perfect replication. This approach necessitates the model prioritizing certain features of the input over others, often leading to the discovery of useful data properties [Goodfellow et al., 2016]. This dimensionality reduction proves beneficial in enhancing classification tasks' efficiency by reducing computational costs and memory overhead, and improving information retrieval in low-dimensional spaces [Goodfellow et al., 2016]. However, traditional autoencoders fall short in generating new data due to the unregulated nature of the latent spaces.

To understand their limitations, consider the analogy of a bag full of numbered balls. If one is tasked with randomly selecting a ball from this bag, the likelihood of picking a specific number is quite low, especially if the bag contains a large number of balls. Each ball in this analogy represents a vector in the latent space of an autoencoder. When an autoencoder extracts features from the input data, the resulting latent space is not perfectly structured; hence, many parts of this space might not correspond to meaningful or recognizable data patterns. This is akin to having many numbers in the bag that do not match the desired number. Thus, when an autoencoder attempts to generate new data by randomly sampling from this unstructured latent space, the likelihood of producing a meaningful output is similarly low. This challenge is further compounded by the fact that the structure of the latent space is influenced by various factors, including the distribution of the source space, the dimensionality of the latent space, and the architecture of the encoder. [Michelucci, 2022].

In essence, traditional autoencoders are not designed to generate new data; their main function is to copy and reconstruct the given input [Goodfellow et al., 2016].

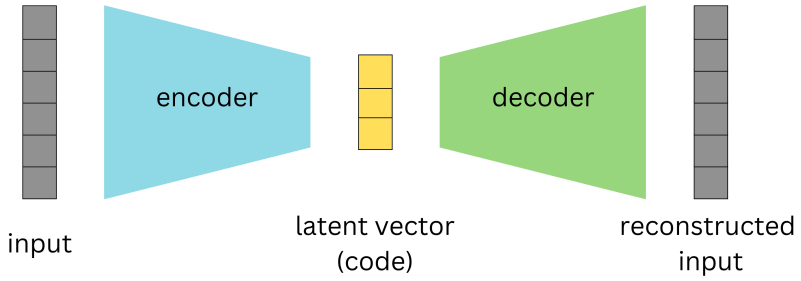


Figure 2: Schematic of an Autoencoder: Demonstrating the process of dimensionality reduction to a latent vector and subsequent reconstruction, aiming to minimize reconstruction loss.

At the core of Variational Autoencoders is the handling of the latent space. This approach addresses the limitations of traditional autoencoders by implementing enhanced regularization within the latent space during training. This is achieved by encoding a distribution over the latent space, characterized by parameters μ (mean) and σ (standard deviation), instead of encoding to a single deterministic point [Doersch, 2016]. A key element in this process is the sampling of a point from this distribution to represent the latent variable. This sampled point is then utilized in the reconstruction of the input data, with the resulting reconstruction error being backpropagated to update the model's weights.

To revisit the analogy of the bag full of numbered balls, VAEs transform the scenario. Instead of a bag with a jumble of numbers, VAEs create a more organized bag where regions are filled with specific ranges of numbers. When a ball (a point in the latent space) is selected at random, there's a higher likelihood of it correlating with a meaningful and recognizable pattern, enabling the generation of coherent outputs. This organization within the latent space is key to VAEs' ability to generate new data.

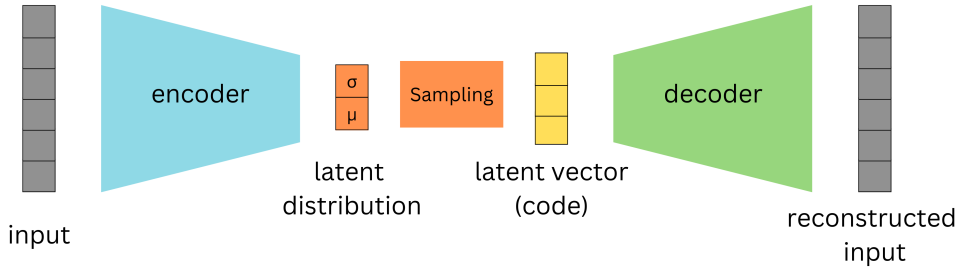


Figure 3: Functionality of a Variational Autoencoder: This figure illustrates the incorporation of a latent distribution, characterized by mean and standard deviation, for enhancing latent space regularization, enabling more effective and diverse data generation.

Despite their advanced capabilities, VAEs have some limitations. A common observation, as noted by Goodfellow et al., is that the generated samples can often appear blurry. This blurriness might stem from the optimization process, specifically the minimization of the Kullback-Leibler divergence. This optimization might lead the model to assign high probabilities to training set points with low 'information value' and other less distinct points, resulting in blurry images [Goodfellow et al., 2016]. The Gaussian distribution often used in VAEs for the generative model may also contribute to this effect, as it can ignore minor features in the input data [Goodfellow et al., 2016]. The performance of the model is also sensitive to the choice of priors for the latent space, making hyperparameter tuning an essential aspect of working with VAEs [Kingma and Welling, 2013; Higgins et al., 2017].

2.2 Generative Adversarial Networks – GANs

"When a deep neural network is used to generate data, the corresponding density function may be computationally intractable" [Goodfellow et al., 2020]. Unlike traditional generative models, implicit generative models do not require the explicit design of a density function to describe the patterns in the data. Instead, they use a sample generation process that produces new samples resembling the existing ones [Goodfellow et al., 2020]. Before Generative Adversarial Networks were introduced, the leading implicit generative model was the generative stochastic network, "which is capable of approximately generating samples via an incremental process based on Markov chains" [Goodfellow et al., 2020]. Markov chains are a way of describing a sequence of events or states, where probability of transition to the succeeding state is solely dependent on current states. This approach, however, can be time-intensive and may not always yield accurate results. GANs, on the other hand, directly generate high-quality samples in a single step, bypassing the gradual and often inefficient process of incremental generation.

The unique adversarial nature of GANs arises from the game-like competition between two neural networks: the generator and the discriminator. The generator is responsible for creating fake inputs or samples, which are then passed to the discriminator. The

discriminator's role is to differentiate between real samples from the domain set and the fake samples generated by the generator.

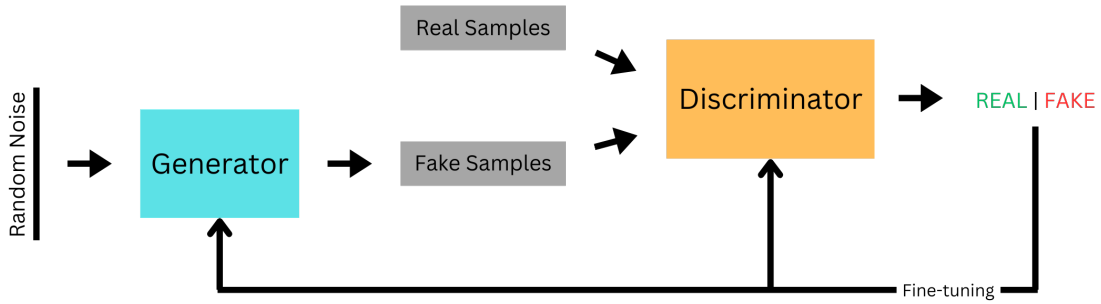


Figure 4: Schematic representation of Generative Adversarial Networks showcasing the interaction between the generator and discriminator networks in generating new data.

In the initial training phase, the discriminator is trained on a dataset of real, unlabeled data, learning to identify the characteristics of authentic samples. As the discriminator becomes adept at recognizing all details, the generator starts creating counterfeits, using random input vectors to produce imitations. The discriminator then evaluates these fakes and provides feedback. This feedback loop, involving sample creation and model adjustments, gradually refines the generator's output. Eventually, the generator becomes so proficient that its fakes are indistinguishable from real data, achieving what is known as a zero-sum game, where one network's gain is the other's loss [Goodfellow et al., 2020].

However, the utility of GANs extends far beyond just image generation. Their applications encompass a wide range of fields, demonstrating their versatility and significant impact. These applications include video frame prediction, which is essential in multimedia applications; image enhancement, crucial in improving the quality and clarity of visual data; and encryption, where GANs contribute to the development of advanced security protocols [Goodfellow et al., 2020]. One of the most notable features of GANs is their ability to learn in an unsupervised manner, particularly through the generator network. Unlike traditional models that require a supervised learning set with labeled data, GANs can generate new data after training the discriminator with real examples. This capability allows GANs to produce realistic and varied outputs without direct exposure to or reliance on a large labeled dataset [Goodfellow et al., 2016],

Nevertheless, GANs pose a substantial challenge in their training process as they are hard to train [Goodfellow et al., 2020]. In addition, Brophy et al. highlight three important problems commonly associated with GANs, among others. These issues, namely non-convergence, diminishing or vanishing gradients, and mode collapse, contribute to the inherent instability experienced during GAN training. Non-convergence refers to the failure of a GAN model to stabilize and reach a state of equilibrium. Instead, it continuously oscillates and fails to converge to a satisfactory solution. As a result, the model does not learn the underlying patterns of the data and can even diverge, leading to poor performance [Brophy et al., 2023]. Diminishing or vanishing gradients occur when the gradients used to update the generator become extremely small or even vanish altogether. This phenomenon is often caused by an overly successful discriminator that becomes too adept at distinguishing real and fake samples. As a result, the generator struggles to learn from the feedback provided by the discriminator, impeding its ability to generate high-quality sam-

ples [Brophy et al., 2023]. Mode collapse happens when the generator collapses, meaning it focuses on producing only a limited set of samples or outputs, typically lacking diversity and variety [Salimans et al., 2016]. In such cases, the generator fails to capture the full range of patterns and characteristics present in the training data, resulting in uniform and repetitive samples that do not adequately represent the true distribution [Brophy et al., 2023].

2.3 Diffusion models

Diffusion models have emerged as a prominent method in generative modeling, offering distinct advantages over traditional models like Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs). They operate by progressively perturbing data with noise and then learning to reverse this process, a methodology that enables the generation of highly realistic and diverse new samples.

Yang et al. [2022] distingueshe between three main approaches that dominate the study of diffusion models, which are going to be discussed shortly: Denoising Diffusion Probabilistic Models (DDPMs) [Ho et al., 2020; Sohl-Dickstein et al., 2015], Score-based Generative Models (SGMs) [Song and Ermon, 2019], and Stochastic Differential Equations (Score SDEs) [Song et al., 2020, 2021]. Each of these models has its own set of strengths and challenges, contributing uniquely to the development of Diffusion Models.

2.3.1 Denoising Diffusion Probabilistic Models

Central to the concept of Denoising Diffusion Probabilistic Models (DDPMs) are two Markov chains: the forward chain and the reverse chain, also known as the forward and reverse diffusion processes [Sohl-Dickstein et al., 2015].

The forward diffusion process, sharing some similarities with VAEs, focuses on a latent feature space of the initial data distribution. However, DDPMs differ in that the forward process in DDPMs "is fixed to a Markov chain that gradually [over a span of T steps] adds Gaussian noise to the data according to a variance schedule $\beta_1, \ldots, \beta_T$" Ho et al. [2020]. This process gradually pertubes the data's structure, eventually resulting in an image of pure noise, with the aim of gradually steering the data distribution towards a more manageable prior distribution [Yang et al., 2022; Poole et al., 2022].

The mathematical formulation of the forward process is given by Martínez et al.:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I) \quad \text{where} \quad \sqrt{1-\beta_t}x_{t-1} = \mu_t \quad \text{and} \quad \beta_t I = \Sigma_t$$

In this equation, the model first adjusts the previous data point x_{t-1} to get x_t, the data point at the current step. This adjustment follows a Gaussian distribution and is done using the term $\sqrt{1-\beta_t}x_{t-1}$, which slightly reduces the intensity or strength of the previous data point [Sohl-Dickstein et al., 2015; Ho et al., 2020]. This controlled approach helps maintain a balance between the original data and the noise, ensuring that the noise doesn't overwhelm the data too quickly. Once this preparatory step is completed, the model then introduces noise. The level of noise added at each step is determined by the parameter β_t, where a higher value means more noise is added [Kingma et al., 2021]. The way noise is added is described by the covariance matrix $\beta_t I$, where I is the identity matrix.

This matrix ensures that noise is added to each element of the data in an independent and uniform manner, evenly distributing the noise across all parts of the data. The importance of this noise-adding process lies in its role in teaching the model the structure and characteristics of noise. By gradually adding noise to the data, the model learns how images degrade step by step, knowledge that is crucial for the reverse process of DDPMs.

The process of adding noise over the entire sequence from the original data point x_0 to x_T is captured another formular by Martínez et al.:

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$$

Built upon the Markov property, the formula implies that each step depends solely on the previous step, allowing for a systematic and gradual transformation from x_0 to x_T [Martínez et al., 2023]. This methodical approach provides a detailed understanding of the data's evolution at each noise addition stage, giving a complete view of the transition probabilities throughout the forward diffusion process.

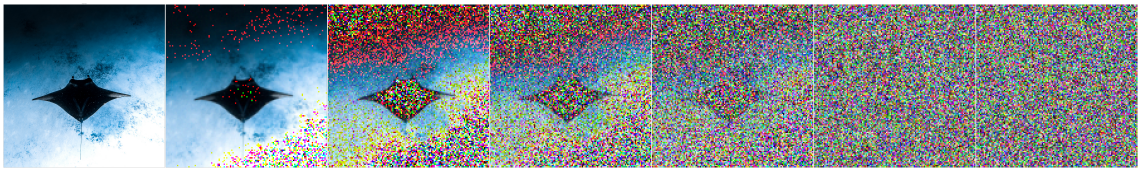


Figure 5: Illustration of the Forward Diffusion Process in DDPMs: This figure demonstrates the gradual addition of Gaussian noise to an image over multiple steps. Each subsequent image from left to right shows an increased level of noise, culminating in the far-right image, which represents a state of pure noise.

The reverse diffusion process employs a neural network parameterized by Θ, to approximate the inverse of the forward process. It estimates the prior state of data points, x_{t-1}, from their current noisy state, x_t, using the probability distribution function $p_\theta(x_{t-1}|x_t)$, as given by Martínez et al.. This process is modeled as a normal distribution where the mean $\mu_\theta(x_t, t)$ and covariance $\Sigma_\theta(x_t, t)$ are determined by the neural network [Yang et al., 2022].

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

$$p_\theta(x_{0:T}) = p_\theta(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t)$$

The latter function $p_\theta(x_{0:T})$ is also taken from Martínez et al. and captures the probability of the entire data sequence under the reverse process, beginning with an estimate of the final noisy data point $p_\theta(x_T)$ and progressively reconstructing the data by removing noise at each step [Ho et al., 2020; Martínez et al., 2023]. Unlike the forward process that adds noise, the reverse process, starting from a state of random noise, uses the learned noise patterns to iteratively generate coherent images. The reverse process is also a Markov chain and involves the neural network learning to predict the reverse diffusion parameters Θ at each timestep [Yang et al., 2022]. The goal here is to ensure that the new samples

it generates are statistically similar to the original data it was trained on. This is done by maximizing the likelihood that these new samples belong to the same overall data distribution as the original set [Yang et al., 2022].

Despite their effectiveness, DDPMs are not without challenges. The most significant of these is the computational time required for generating new samples, which is due to "a Markov process [that] has to be simulated at each generation step, which greatly slows down the process" [Martínez et al., 2023].

2.3.2 Score-Based Generative Models

In the domain of generative modeling, Score-Based Generative Models (SGMs), as introduced by Song and Ermon, distinguish themselves by prioritizing the learning of a score function using the Stein score [Liu et al., 2016]. The score is "the gradient of the log-density function at the input data point" [Song and Ermon, 2019], serving as a guide towards higher data density regions. The score function is central in SGMs, akin to how DDPMs use a forward process of adding noise, but with a different purpose. In SGMs, the score guides the model to areas in the data space where the probability of finding similar data points to the training set is higher.

SGMs begin their learning process with a set of training data. Common data distributions indicate areas of high probability, while rare inputs usually result in low represented data regions. The role of SGMs is to learn how to work with that data space. In practice, the model $s_\theta(x)$ is trained to replicate the score function $\nabla_x \log p(x)$ of a data distribution $p(x)$ using so called score matching [Hyvärinen and Dayan, 2005]. Training of such models is achieved by "minimizing the Fischer divergence between the model and the data distributions" Song, denoted as:

$$\mathbb{E}_{p_{(x)}} \left[\|\nabla_x \log p(x) - s_\theta(x)\|_2^2 \right]$$

This divergence assesses how closely the model's score aligns with the actual data score by calculating their squared differences. The integration of score matching into this process means that no precise knowledge of the true data value is required [Song, 2021]. The goal here is to align the model's predictions with the real data's log-likelihood gradient as closely as possible. Under specific conditions, this method can reliably ensure that the model's score accurately reflects the true score of the data distribution [Song and Ermon, 2019].

In the generation of new samples, SGMs start with a random or noise image. Using Langevin dynamics, an iterative process described by [Roberts and Tweedie, 1996], the model samples from a distribution $p(x)$ using its score function. Following Song, it starts with a random prior distribution $x_0 \sim \pi(x)$, and iteratively updates the chain by:

$$x_{i+1} \leftarrow x_i + \epsilon \nabla_x \log p(x) + \sqrt{2\epsilon} z_i,$$

where each iteration updates the sample x_i by a small step size ϵ in the direction of the data's score function, guiding the sample towards the target data distribution. This process theoretically converges the distribution to $p(x)$ as ϵ becomes smaller and iterations increase [Song and Ermon, 2019; Song, 2021].

A notable challenge in SGMs arises when estimating the score function in less represented data regions. Inaccuracies here may hinder the convergence of Langevin Dynamics

[Song and Ermon, 2019]. To mitigate this, Song and Ermon recommend introducing Gaussian noise to the data and estimating scores for these noise-altered distributions, an approach termed Noise Conditional Score Networks (NCSNs). This strategy prevents the data distribution from collapsing into a lower-dimensional structure, ensuring more robust sample generation [Song and Ermon, 2019].

The core idea of Noise Conditional Score Networks is to generate realistic data samples by effectively navigating through a series of noise-perturbed data distributions that gradually converge to the true data distribution. This process is facilitated by introducing Gaussian noise σ into the data at multiple levels [Song and Ermon, 2019]. The NCSN, represented $s_\theta(x, \sigma)$, is trained to estimate scores for these noise-perturbed distributions. This training allows the network to adjust its predictions based on different noise levels to ensure accurate and relevant point estimates. Generating new samples begins with a state of pure noise and uses annealed Langevin dynamics in order to gradually reduce the noise level σ, ending when a sufficiently low noise level yields a sample that accurately represents the modeled data [Song and Ermon, 2019].

2.3.3 Score-Based Generative Modeling through Stochastic Differential Equations – SDEs

Song et al. aim to combine both DDPMs and SGMs (NCSNs) using Stochastic Differential Equations (SDEs). The idea is to create a continuous diffusion process, indexed by time, that transforms a data distribution into a more tractable prior distribution. This is described through the following function by Song et al.:

$$dx = f(x, t)dt + g(t)dw,$$

The process is governed by two coefficients: a drift coefficient $f(x, t)$, governing the deterministic properties of the stochastic process, guiding how data evolves over time, and a diffusion coefficient $g(t)$, which scales the random noise introduced by Brownian motion dw (Wiener process) [Song et al., 2020]. This Brownian motion represents the random movement of particles in a fluid as they collide with fast-moving molecules in the fluid.

For generating new samples, a principle from Anderson [Anderson, 1982] comes into play. It states that "the reverse of a diffusion process is also a diffusion process, running backwards in time and given by the reverse-time SDE:" [Song et al., 2020].

$$dx = \Big[f(x, t) - g(t)^2 \nabla_x \log p_t(x) \Big] \, dt + g(t) d\bar{w}$$

This equation by Song et al. describes the process of recovering data from noise by moving backward in time. "Once the score of each marginal distribution, $\nabla_x \log p_t(x)$, is known for all t, we can derive the reverse diffusion process from [the above equation] and simulate it to sample from p_0" [Song and Ermon, 2019]. This score function essentially captures the essence of the data's probability distribution at various stages of noise addition.

Training a model to accurately estimate the score functions at various noise levels is a fundamental aspect of this process. To achieve this, the model is trained through score matching [Hyvärinen and Dayan, 2005], which involves fine-tuning the model to closely approximate these score functions across a spectrum of noise levels [Song et al., 2020].

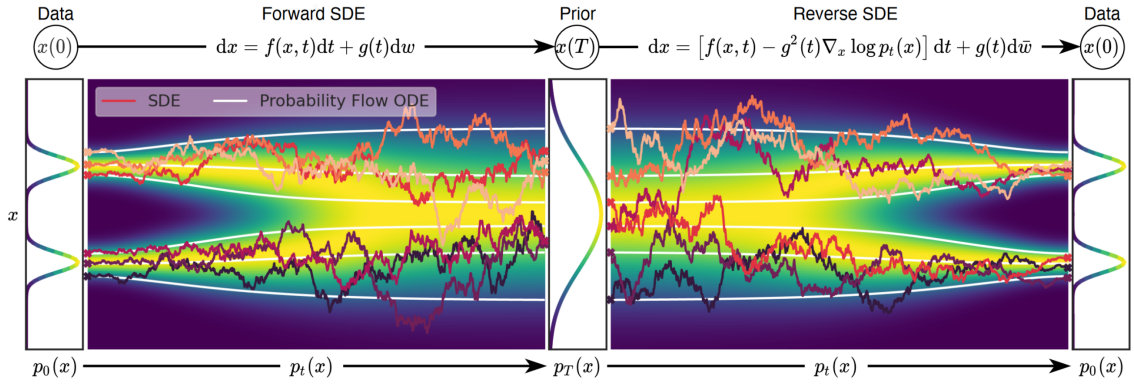


Figure 6: This figure, adapted from Song et al. [Song et al., 2020], illustrates the two-fold process in score-based generative modeling through SDEs. On the left, the Forward SDE represents the gradual transformation of data into noise, guided by the drift and diffusion coefficients. On the right, the Reverse SDE depicts the process of reconstituting original data from noise, leveraging the known score of each marginal distribution.

The score matching process matches the output of the score network with the true gradient of the log-likelihood over the course of the SDE, enabling the generation of realistic data samples from complex distributions.

2.4 Contrastive Language-Image Pre-training – CLIP

Radford et al. address the limitations of traditional computer vision models, which are restricted by their training on a fixed set of object categories and lack adaptability to new tasks or concepts. To overcome these limitations, they propose a novel method called Contrastive Language-Image Pre-training (CLIP), which is an "efficient and scalable method of learning from natural language supervision" [Radford et al., 2021]. This process allows the model to learn a representation of the image that is grounded in natural language, enabling it to understand the content and context of the image.

Unlike traditional computer vision models that rely solely on annotated image datasets, CLIP leverages lots of text and image pairs from the internet. It learns to associate images and their corresponding textual descriptions, allowing it to understand the relationship between visual and textual data.

One of the remarkable aspects of CLIP is the ability for zero-shot capability. It can perform tasks without task-specific training [Radford et al., 2021]. For example, given a natural language prompt, CLIP can recognize objects in images, generate captions, or perform classification tasks.

However, there exist several limitations to CLIP. "The performance of zero-shot CLIP is often just competitive with the supervised baseline of a linear classifier on ResNet-50 features" [Radford et al., 2021]. This means that CLIP is not significantly better than a model that is trained on labeled data for the specific task at hand.

2.5 Stable Diffusion

Stable Diffusion [Rombach et al., 2022] is a type of latent diffusion model specifically designed to generate AI images from text prompts. It operates by compressing images into a latent space, which is substantially smaller than the high-dimensional image space, making the process faster and more efficient.

The core of Stable Diffusion lies in its diffusion process, which includes both forward and reverse diffusion. The model uses a Variational Autoencoder (VAE) to manage the compression and decompression of images into and from the latent space. Training involves teaching a noise predictor, part of the U-Net model, to estimate the noise added at each step of the forward diffusion process. This predictor is then used in reverse diffusion to subtract the estimated noise from the noisy images, thus recreating a clear image.

Conditioning is an important aspect of Stable Diffusion, especially for text-to-image generation [Rombach et al., 2022]. Text prompts are tokenized by CLIP and converted into embeddings, which are then processed by a text transformer. This processed data steers the noise predictor, influencing the image generation process to align with the textual input.

Stable Diffusion also includes functions like image-to-image transformation, inpainting, and depth-to-image generation, each with specific steps and mechanisms. CFG (Classifier-Free Guidance) value [Ho and Salimans, 2022] is an important parameter that determines how closely the generated image should follow the text prompt.

2.6 Multilayer Perceptron – MLP

Multilayer Perceptrons (MLPs) form a fundamental class of artificial neural networks in machine learning, characterized by their layered structure, including input, hidden, and output layers [Noriega, 2005; Murtagh, 1991]. Central to their function is the activation function, typically the Logistic Sigmoid Function, which transforms the weighted inputs into node outputs in a continuous and differentiable manner, facilitating gradient-based optimization [Murtagh, 1991; Noriega, 2005]. The learning process in MLPs is supervised, involving initial random weight assignment, followed by training through pattern presentation, output comparison, and backward error propagation, typically using gradient descent methods to minimize errors [Noriega, 2005]. This process is guided by the generalized delta rule or backpropagation, which adjusts network weights based on the error between predicted and actual outputs, allowing the network to effectively learn from its environment [Murtagh, 1991]. The architecture of an MLP, including the number of layers and nodes, along with the activation methods and learning techniques, significantly influences its performance. MLPs are versatile, capable of performing tasks like regression, mapping input vectors to values, and supervised classification, where input patterns are trained to produce specific output classifications [Murtagh, 1991]

2.7 Representation Forms of 3D Data

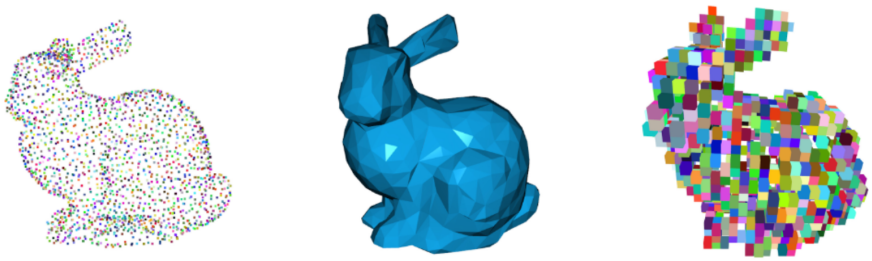
2.7.1 Meshes, Point-Clouds and Voxels

Meshes – A mesh is a geometric data structure used to represent the surfaces of 3D objects through subdivisions into a network of polygons. These polygons are typically formed by

vertices connected by edges, creating faces or facets. The most common form of meshing is triangular meshing, where each face is a triangle. Meshes are pivotal in computer graphics and 3D modeling for their ability to represent complex surfaces efficiently, facilitating accurate rendering and transformation [Lahav and Tal, 2020; Zhang et al., 2023].

Point-Clouds – A point cloud is a collection of data points in a 3D coordinate system (X, Y, Z). They offer quick rendering and transformation capabilities, which is advantageous for direct inspection. However, point clouds may not integrate seamlessly into sophisticated 3D applications that primarily use mesh-based rendering [Xu et al., 2021; Zhang et al., 2023].

Voxels – A voxel is analogous to a 3D pixel, representing the smallest unit in a voxel-based model - a collection of 3D pixels. It is a discretized model primarily associated with solid modeling. In point cloud data, voxels can be used to represent each point, providing a filled view of the spaces between points [Xu et al., 2021; Zhang et al., 2023].



(a) Point Cloud (c) Mesh (b) Voxel

Figure 7: Representation forms of 3D data, adapted from Zhang et al. [Zhang et al., 2023].

2.7.2 Neural Radiance Fields – NeRFs

Neural Radiance Fields (NeRFs), as introduced by Mildenhall et al., represent a significant advancement in 3D scene representation, particularly when compared to previous methods which often struggle with complex geometries and varying lighting conditions. Emerging in response to these challenges, NeRFs employ a novel volumetric representation, capturing the spatial and angular distribution of light more accurately [Mildenhall et al., 2021].

NeRFs start the generation process by initializing a 3D scene with random values. This serves as a starting point for refining the scene through training a Multilayer Perceptron (MLP) F, parameterized by θ. The MLP consists of Fully Connected Layers (FCs) with ReLU activations, specifically designed to encode the volumetric details of that particular scene, effectively creating a dedicated neural network for each scene [Mildenhall et al., 2021]. The neural network accepts two types of input: a position in a given coordinate system, expressed as a 3D vector x, y, z, and a viewing direction represented by two angles θ, ϕ. Imagine a scenario where a flashlight is held in the middle of a dark room. The angle θ would represent how much the flashlight is tilted up or down. Similarly, the angle ϕ would represent how much the flashlight is rotated about the vertical axis while pointed outward. The network's output consists of the color c and density σ at that particular location [Mildenhall et al., 2021].

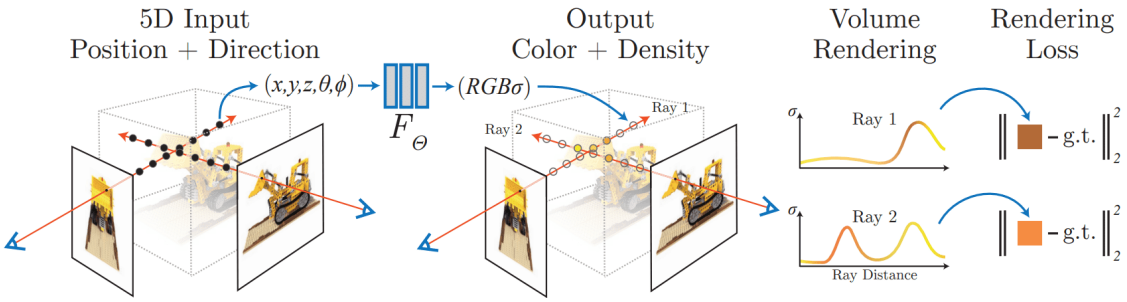


Figure 8: Sumamrized workflow of a NeRF as shown in [Mildenhall et al., 2021]: 5D input (Position + Direction) is processed by the MLP F_θ to output color and density. Volume rendering integrates predictions along rays to generate an image from the specified viewpoint. The rendering loss, comparing the rendered and actual images, guides the network's training and refinement process.

Central to NeRF's operation is the process of volume rendering, which involves casting rays through the scene and gathering color and density information at multiple points.

$$C(r) = \int_{t_n}^{t_f} T(t)\sigma(r(t))c(r(t), d)dt \quad \text{where} \quad T(t) = \exp\left(-\int_{t_n}^{t} \sigma(r(s))ds\right)$$

This formula, as outlined by Mildenhall et al., enables the rendering of a 3D scene by casting rays and aggregating their properties from a near boundary (t_n) to a far boundary (t_f). The equation essentially builds up an image by integrating the effects of light interaction with the scene's material over the length of each ray. The volume density σ at a point in the scene indicates the amount of light-blocking material at that specific location. A higher density suggests a greater likelihood of a light ray being obstructed, signifying more material at that point. The accumulated transmittance $T(t)$ reflects the cumulative effect of density along a ray's path. It quantifies the extent to which light from the start of the ray can travel through the scene to a given point without being absorbed or scattered by the material. Its value decreases along the ray's path as it encounters areas of higher density. The point at which this density curve rises significantly usually corresponds to an object in the scene, and the color $c(r(t), d)$ at this point is what is rendered for that particular pixel. As seen in Figure 8, these density and color curves can be visualized to illustrate how density and color vary along the ray's path [Mildenhall et al., 2021]. Repeating this ray casting process for every pixel of an image effectively creates $C(r)$, a 2D "screenshot" of the current 3D scene, given the viewing direction d of where each ray is shot.

Classical deep learning methods often require a comprehensive dataset of 3D models comprising different scenes and their representations. NeRFs overcome this limitation and rely only on a set of 2D images of the target object from multiple angles that serve as ground truth. After volume rendering, NeRFs compare the newly generated image of the scene with the corresponding real image from the original 2D set. The difference between these images is calculated using the total squared error, which determines the difference between the original and the rendered image [Mildenhall et al., 2021]. This loss is then backpropagated to the MLP, which adjusts its parameters in response to this calculated

loss aiming to reduce the discrepancy between both images. This iterative adjustment process is applied to all viewing angles of the original data set until the MLP reaches convergence, where the rendered images closely match the real images.

To address multi-view consistency, NeRF predicts color as a function of both location and viewing direction, while density depends solely on location. This separation acknowledges that density, unlike color, remains consistent regardless of viewing angle [Hu et al., 2023].

Although naive NeRF models may not provide photorealistic results due to the lack of detail, several optimizations have been introduced to improve their performance. One notable improvement is the use of positional encoding techniques that deterministically map 3D coordinates and view directions to a higher dimensional space. This is achieved by using high-frequency features before inputting them to the multilayer perceptron (MLP), which helps optimize the neural radiation fields to better represent high-frequency scene content [Mildenhall et al., 2021]. Hierarchical volume sampling offeres another optimization. This strategy involves two neural network systems, one coarse and one refined, which are jointly optimized during training. Initially, the coarse network sparsely samples the rays in the 3D scene, and based on this initial sampling, a refined network is guided to perform a more detailed sampling. This hierarchical approach is beneficial because dense sampling is very computationally expensive. A two-tier system therefore helps manage computational resources while allowing detailed sampling along rays to obtain the density and color of the sampled points [Arandjelović and Zisserman, 2021].

One of NeRFs key advantages is its memory efficiency. For example, rendering a single scene with NeRF requires only about five megabytes of memory, which is in stark contrast to voxel grid renderings that require over 15 gigabytes for a comparable scene [Mildenhall et al., 2021]. This mismatch in memory requirements underscores NeRF's superior efficiency in terms of data storage and transfer, and represents a compelling advantage over traditional 3D rendering techniques [Mildenhall et al., 2021]. Remarkably, the memory requirement of the rendered scene is even smaller than that of the input images, making the model extremely efficient in data storage and transfer [Mildenhall et al., 2021].

However, the NeRF model is not free of limitations. A major challenge is the computational cost associated with training the neural network. The optimization process for a single scene may require about 100,000 to 300,000 iterations, equivalent to a training period of about one to two days, to converge, assuming the use of a single NVIDIA V100 GPU [Mildenhall et al., 2021]. In addition, NeRF rendering is prone to sampling and aliasing issues that can lead to significant artifacts in the synthesized images [Rabby and Zhang, 2023]. These artifacts arise from the limited sampling of the radiation field, resulting in inaccurate reconstruction of certain features, especially in scenes containing sharp edges or textures [Rabby and Zhang, 2023].

2.7.3 Deep Marching Tetraheda – DMTet

In 3D model generation, there are two primary methods: implicit and explicit 3D representations [Shen et al., 2021]. Implicit methods, like Neuronal Radiance Fields (NeRFs) [Mildenhall et al., 2021], use mathematical functions to define shapes, capturing complex details such as the shape's orientation and volume. These representations, while rich in detail, usually require conversion into 3D meshes for practical use. Conversely, explicit

methods involve directly defining 3D shapes using specific coordinates, like in the case of meshes. This approach is able to capture and show geometrical details. However, it can be less ideal for computational tasks, especially in machine learning, due to its irregularities [Michalkiewicz et al., 2019].

DMTet, or Deep Marching Tetrahedra [Shen et al., 2021], emerges as a method representing a pivotal step in converting neural implicit representations into explicit mesh forms. Unlike traditional approaches that often struggle with detailed 3D structures, DMTet leverages advanced algorithms to produce high-fidelity 3D meshes in "a new differentiable shape representation [...]" [Shen et al., 2021]. DMTet uniquely combines the benefits of both implicit and explicit 3D representations, leveraging a novel hybrid 3D representation approach.

The method propesed by Shen et al. produces a deformable and differentiable tetrahedral grid based on a given Sign Distance Field (SDF). A SDF is a function defined over a 3D space, where each point in the space gets a value that represents its shortest distance to a surface. The value is negative if the point is inside an object and positive if it's outside. The first step is therefore always to convert the original input into such a form. Point clouds are transformed directly, while voxels undergo initial conversion into point clouds through sampling points on their surfaces, which can then be used for the calculation. In the tetrahedral grid, each vertex receives an initial predicted SDF value and a deformation offset to represent the surface using an implicit function [Shen et al., 2021]. The surface is then refined using subdivision and further "converted into an explicit [triangular] mesh with a Marching Tetrahedra (MT) algorithm, which we show is differentiable and more performant than the Marching Cube" [Shen et al., 2021]. The final step involves refining the mesh into "a parameterized surface with a differentiable surface subdivision module" [Shen et al., 2021].

DMTets allows for supervision on the surface which produces better results in contrast to NeRFs [Shen et al., 2021]. Moreover, it is efficient in terms of inference speed and output quality, producing explicit meshes suitable for interactive graphic applications [Shen et al., 2021].

2.7.4 Instant Neual Graphics Primitives

Müller et al. presents Instant NGP, a technique offering advancements in neural rendering primarily through its efficient encoding techniques and optimized rendering processes. This method is applied across multiple tasks, including Gigapixel Image, Neural Signed Distance Functions (SDF), Neural Radiance Caching (NRC), and Neural Radiance and Density Fields (NeRF) [Müller et al., 2022]. InstantNGP should not be viewed directly as a distinct representation type for 3D models, but rather as a significant extension of existing methods, especially in the context of NeRFs, which are relevant in this thesis.

InstantNGP utilizes multi-resolution hash encoding, efficiently mapping "neural network inputs to a higher-dimensional space" [Müller et al., 2022]. This process includes hashing, feature vector lookup, linear interpolation, and concatenation with viewing direction parameters, crucial for rendering RGB and density predictions effectively [Müller et al., 2022]. A significant innovation in InstantNGP is its enhanced ray marching algorithm, combined with an occupancy grid. This grid improves rendering by skipping non-contributing spaces, thus enhancing training times drastically [Müller et al., 2022].

The model architecture in InstantNGP varies depending on the task. For NeRFs, the

architecture involves two concatenated Multilayer Perceptrons (MLPs), a density MLP, mapping the hash-encoded position to output values, and a color MLP, which adds view-dependent color variation, for volumetric representations [Müller et al., 2022].

InstantNGP opts for concatenation over reduction in its encoding process to maintain the richness of encoded information and facilitate parallel processing of each resolution [Müller et al., 2022]. However, a challenge is the microstructure due to hash collisions, which leads to a "grainy" appearance [Müller et al., 2022]. Overcoming this issue, potentially through advanced filtering or smoothness techniques, is identified as a key area for further improvement [Müller et al., 2022].

Chapter 3

Models

This chapter presents an in-depth exploration of various models used in the automatic generation of 3D models. It bridges the theoretical foundations laid in the previous chapters with practical applications and technologies that are developing the field of 3D modeling. Furthermore, it delves into different categories of model generation, specifically focusing on 3D models generated from text and images. This comprehensive analysis not only highlights the diversity and complexity of current methods but also underscores the rapid advancements and potential future developments in this domain. The below timeline shows the chronological development of these models, offering a perspective on how the field has progressed over time.

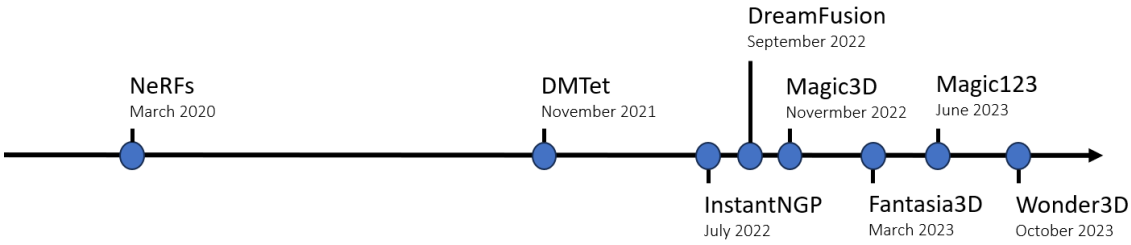


Figure 9: Timeline of generative 3D modeling technologies: This figure outlines important milestones in the development of the key methods discussed in this thesis.

The chapter is structured to critically examine the methodologies, strengths, and limitations of each model. This detailed exploration provides an in-depth understanding of the current state of 3D model generation, emphasizing both the technological intricacies and their practical implications.

3.1 3D from Text input

The ability to create 3D models from text opens up a wide range of possibilities. For designers, artists and architects, this means a more intuitive and accessible way to bring ideas to life. This section will focus on how these systems interpret text descriptions and explain the computational challenges they face.

3.1.1 Dreamfusion

DreamFusion, as introduced by Poole et al., marks a significant advancement in the field of 3D modeling. Utilizing Neural Radiance Fields (NeRF), DreamFusion employs a novel technique known as Score Distillation Sampling (SDS) to generate coherent 3D objects and scenes from a variety of text prompts. This approach diverges from traditional methods that depend on pre-existing images from multiple angles, as DreamFusion dynamically generates these images during training using a 2D diffusion model.

Central to DreamFusion is the use of Differentiable Image Parameterization (DIP), as described by [Mordvintsev et al., 2018]. This technique enables the generation of images x through parameters θ and a differentiable generator g, offering refined optimization capabilities even at the pixel level [Poole et al., 2022]. This marks a shift from the conventional approach of diffusion models, which usually produce outputs similar to their training data. In DreamFusion, the parameters θ define 3D volumes, with g functioning as a volumetric renderer.

This method offers an innovative approach to transforming text into 3D models, as depicted in Figure 10. This process employs several key components: a text prompt with the desired goal that serves as a guide, Google's Imagen as the text-to-image diffusion model [Saharia et al., 2022], an imporved version of Neural Radiance Field, the mip-NeRF 360 [Barron et al., 2022] "that reduces aliasing" [Poole et al., 2022], and the Score Distillation Sampling (SDS) for the loss function.

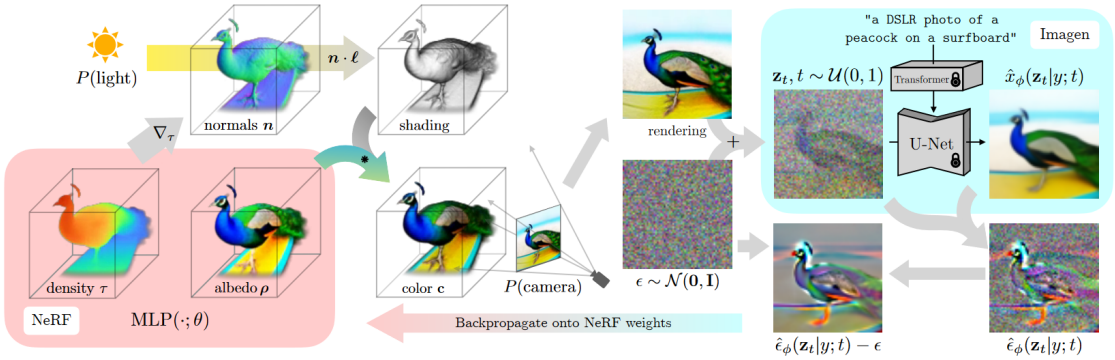


Figure 10: Overview of DreamFusion's process for transforming text into 3D models, illustrating the integration of Neural Radiance Fields, Score Distillation Sampling, and diffusion models. Image adapted from Poole et al. [Poole et al., 2022].

At the center of the rendering process is the Neural Radiation Field, which is represented as $\text{MLP}(\cdot;\theta)$, where the dot $\cdot$ denotes the input of 3D coordinates and viewing directions. This input is important to determine how light and color interact in 3D space. θ symbolizes the parameters of the MLP that are fine-tuned during training. These parameters determine how the MLP interprets its input (the 3D coordinates and viewing directions) to produce the final output, such as the color and density at each point in the 3D model.

The process of generating 3D scenes in DreamFusion begins with the random initialization of the NeRF MLP's parameters. This is followed by the selection of random camera angles and lighting positions, ensuring realistic and varied perspectives [Poole et al., 2022]. Multiplying the light position l by the normals derived directly from the density gradients

calculated during NeRF training produces the shading output, an essential element for realism. Additionally, the inherent color of objects, or albedo, is also generated during the NeRF rendering phase. By combining albedo and shading effects, the NeRF can render accurate colors for every point in the scene, resulting in detailed visual representations from any chosen viewing angle. Following these steps, an image is sampled from the current state of the NeRF representation using ray casting for every pixel.

Unlike traditional NeRFs, which rely on the loss of total squared error for training, DreamFusion uses Score Distillation Sampling (SDS) to refine the model. SDS differs from the total squared error approach in that the pixel values are not compared directly. Instead, the model initially adds random noise ϵ to its rendering, based on the current timestep t, simulating a 'diffusion' process [Poole et al., 2022]. It then uses the Imagen diffusion model to evaluate the diffused rendering z_t based on its knowledge of what makes an high quality image given a text prompt y. Imagen aims to predict the denoised version of the image $\hat{x}_\phi(z_t|y;t)$ of the previous timestep.

The next step involves combining the original diffused rendering with the denoised prediction of Imagen in order to copmute the noise content $\hat{\epsilon}_\phi(z_t|y;t)$. Finally, the initially added noise ϵ is substracted from the noise content $\hat{\epsilon}_\phi$, resulting in a low variance update direction [Poole et al., 2022]. Essentially, this is a clearer direction of how the NeRF should change to produce an image that better matches the text prompt. This update direction is then backpropagated to the NeRF MLP, which allows its parameters θ to be fine-tuned accordingly.

The primary SDS function, as formualted by Poole et al., is denoted as:

$$\nabla_\theta \mathcal{L}_{\text{SDS}}(\phi, \mathbf{x} = g(\theta)) \triangleq \mathbb{E}_{t,\epsilon}\left[w(t) \left(\hat{\epsilon}_\phi(\mathbf{z}_t; y, t) - \epsilon\right) \frac{\partial \mathbf{x}}{\partial \theta} \right]$$

In this equation, $w(t)$ acts as a weighting factor, modifying the influence of different components within the formula. Additionally, the term $\frac{\partial \mathbf{x}}{\partial \theta}$ indicates how changes in the model's parameters θ affect the image $\mathbf{x}$, guiding the optimization process.

Despite DreamFusion's promising results, it is not without limitations. The model tends to exhibit a lower level of detail, partly due to its reliance on a 64×64 image model [Lin et al., 2023]. Furthermore, while SDS is an effective loss function, it sometimes leads to "oversaturated and oversmoothed results [...]" [Poole et al., 2022]. Another aspect to consider with SDS is the mode-seeking behavior, which potentially limits the variety of results generated. This limitation is strengthened by the use of KL divergence, "which has been previously noted to have mode-seeking properties in the context of variational inference and probability density distilaltion"[Poole et al., 2022]. This tendency of the model to prioritize the most frequent patterns can lead to a trade-off between accuracy and diversity, so "it may be unclear if minimizing this loss will produce good samples" [Poole et al., 2022]. This statement highlights a major challenge in machine learning, especially with generative models such as DreamFusion. Minimizing loss, a standard method for improving model performance, does not always lead to high-quality or diverse results. There is a risk of overfitting, where the model can reproduce the training data very well, but is less able to generate new and diverse results.

3.1.2 Magic3D

Magic3D represents a significant advancement in the domain of high-resolution 3D model generation from text input. Developed by Lin et al., this method overcomes limitations observed in previous models like DreamFusion, particularly in terms of optimization speed and resolution of the generated models.

The core technique employed by Magic3D involves a coarse-to-fine optimization process. Initially, a coarse model is generated using a low-resolution diffusion prior, which is then fine-tuned in the second stage to yield a high-quality textured 3D mesh model [Lin et al., 2023]. This approach allows Magic3D to generate detailed 3D models with enhanced texture quality in a comparatively shorter time frame.

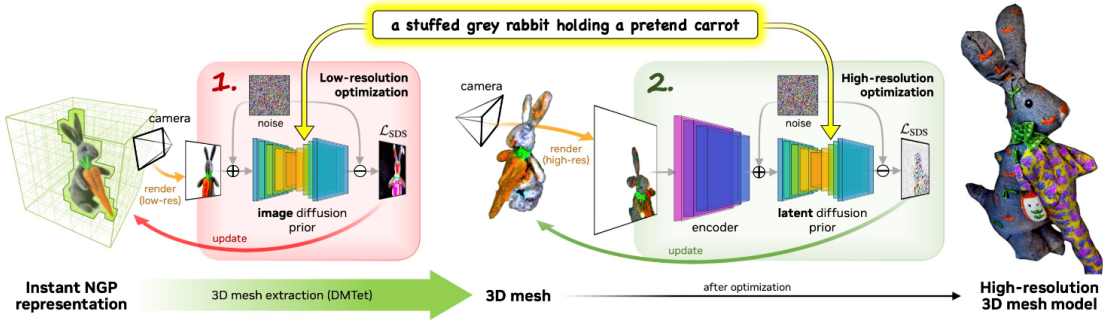


Figure 11: This illustration from [Lin et al., 2023] showcases the Magic3D process, beginning with the InstantNGP for initial 3D representation. It then details the coarse-to-fine procedure, evolving to a refined high-resolution 3D mesh model.

Magic3D's process of generating 3D models from text prompts begins with an initial stage that employs the eDiff-I base diffusion model [Balaji et al., 2022], akin to Google's Imagen [Saharia et al., 2022]. This model operates at a relatively low resolution of 64×64, laying the groundwork for the 3D geometry and textures given from the text input [Lin et al., 2023]. During this phase, the model uses a sparse 3D hash grid structure from Instant NGP [Müller et al., 2022], "which allows us to represent high-frequency details at a much lower computational cost" [Lin et al., 2023]. The optimization in this phase is performed by two single-layer neural networks that predict albedo (the base color of the object), density (how solid or transparent parts of the object are), and normals (which determine how light bounces off the surface) of the object [Lin et al., 2023]. This approach not only speeds up the optimization process but also ensures the foundational quality of the coarse 3D model [Lin et al., 2023]. This stage involves sampling an image from the NGP representation, which is then infused with an initial state of noise through the image diffusion prior. This step sets the stage for iterative refinement. The refinement is guided by the Score Distillation Sampling (L_{SDS}) loss function, which allows the model to iteratively improve its accuracy. The use of the image diffusion prior in this low-resolution phase is primarily for shaping the overall structure and layout of the model, focusing more on broad features rather than fine details.

In the refinment stage, Magic3D focuses on refining this coarse model into a high-resolution textured mesh, marking a transition from the basic structure to a detailed

representation. The refinement process starts with extracting a 3D mesh with the help pf DMTet, where each grid vertex contains a value indicating its distance from the surface (signed distance field) and its deformation from the original position [Shen et al., 2021; Lin et al., 2023]. The process begins by sampling a new image from this 3D mesh. This image is then processed by an encoder, which converts it into a format suitable for further processing by the latent diffusion model. The latent diffusion model used here, particularly Stable Diffusion [Rombach et al., 2022], operates at a high resolution of 512×512, allowing for more detail of the final model [Lin et al., 2023]. Once the latent diffusion prior has processed the image, its output is evaluated using the Score Distillation Sampling L_{SDS} loss function. This function compares the generated image with a target image, enabling the model to fine-tune each vertex's signed distance field and deformation. A technique employed here is the increase in focal length to zoom in on object details, essential for recovering high-frequency details in both geometry and texture [Lin et al., 2023].

Magic3D's proficiency is not limited to just creating high-quality 3D models; it also offers extensive creative control over the generation process. The model supports "fine-tuning a learned coarse model with a new prompt" [Lin et al., 2023], allowing users to influence the generated output significantly. This includes text-based edits and image-conditioned generation, increasing the scope for creative and personalized 3D model generation [Lin et al., 2023].

3.1.3 Fantasia3D

The model proposed by Chen et al. takes a different approach to generating 3D models from text input, in particular by disentangling geometry and appearance in the generated 3D models. This method offers a more detailed rendering quality compared to conventional Neural Radiance Fields (NeRFs), which use volume rendering to combine the learning of surface geometry with pixel colors. This conventoional approach limits effective surface recovery, lacking the capability to track the surface of an object and tune detailed material and texture. In contrast, Fantasia3D achieves more realistic outputs with its hybrid scene representation of DMTet, "which maintains a deformable tetrahedral grid and a differentiable mesh extraction layer; deformation can thus be learned through the layer to explicitly control the shape generation" [Chen et al., 2023]

In the geometry stage, Fantasia3D relies on a Deformable Mesh Tetrahedralization (DMTet), which parametrizes the 3D geometry as a Multi-Layer Perceptron (MLP) Ψ. Initially, Fantasia3D renders and encodes surface normals and object masks extracted from DMTet. However, in later stages, the model refines its approach by using only the rendered normal map for shape encoding [Chen et al., 2023]. The default initialization of DMTet is an ellipsoid, but the model also accepts custom inputs.

In the Appearance stage of Fantasia3D, the aim is to achieve photorealism in surface rendering using the DMTet from the geometry stage, by integrating the Physically-Based Rendering (PBR) material model [McAuley et al., 2012]. This model includes the diffuse term k_d, the combined roughness and metallic term k_{rm}, and the normal variation k_n [Chen et al., 2023]. To predict these material parameters, a Multilayer Perceptron (MLP) denoted as Γ is used. The process involves the application of the formula $(k_d, k_{rm}, k_n) = \Gamma(\beta(p); \gamma)$ by Chen et al., where $\beta(p)$ represents the surface properties at a specific point p and γ indicates the network parameters. The rendering of each pixel's color within this framework involves the integration of incident light and the Bidirectional Reflectance

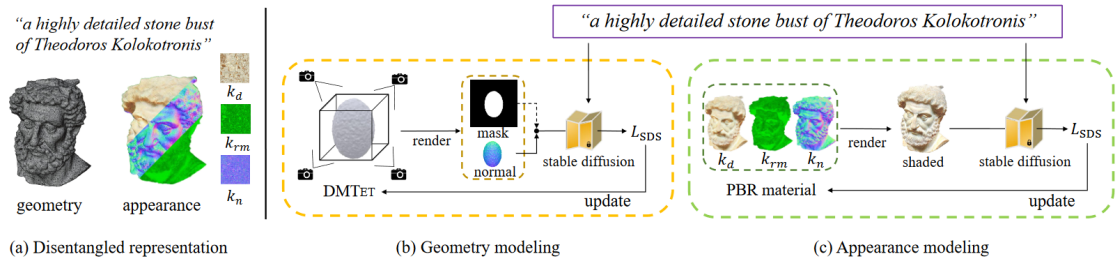


Figure 12: Overview of Fantasia3D's workflow as given in [Chen et al., 2023], disentangling geometry from appearance modeling and iteratively enhancing the quality using a refinment process.

Distribution Function (BRDF) [Chen et al., 2023]. The BRDF plays a crucial role in accurately depicting how light reflects off different surfaces, with its parameters directly influenced by the material properties (k_d, k_{rm}, k_n). Additionally, the appearance generated in this stage is converted into 2D texture maps. These maps are aligned with the UV maps produced, ensuring a seamless and realistic texturing of the 3D model surfaces.

Both MLPs (Γ for appearance and Ψ for geometry) undergo a refinement process, using a pre-trained Stable Diffusion model [Rombach et al., 2022]. This model improves the capabilities of Γ and Ψ, ensuring that they accurately interpret and render 3D shapes and textures. A key aspect of this refinement is the use of Score Distillation Sampling (SDS) loss [Mildenhall et al., 2021] for optimization. This step allows the model to evaluate and adjust its rendering based on how accurately it replicates the true image of Stable Diffusion. By iterating this process vor multiple viewpoints, the model continually improves its accuracy in rendering photorealistic images, ensuring that the final output is as close to the actual image as possible.

Fantasia3D offers a high degree of user interactivity, permitting the incorporation of both custom and predefined generic 3D shapes, thus greatly enhancing the versatility and user engagement in the content creation process. The separation of geometry and appearance generation also ensures compatibility with widely-used graphics engines [Chen et al., 2023]. Despite its capabilities in creating high-quality 3D models from textual descriptions, Fantasia3D encounters specific challenges. One notable limitation is its struggle with accurately generating complex geometries like hair, fur, and grass [Chen et al., 2023]. Furthermore, the model is currently not able to generate complete scenes as focus is currently lying on individual object generation [Chen et al., 2023].

3.2 3D from Image

The conversion of two-dimensional images into three-dimensional models represents a significant challenge in the domain of computer vision and 3D modeling. This section explores the methodologies and technologies employed in transforming 2D images into 3D models, a process that has profound implications in various fields, including virtual reality, gaming, and medical imaging. Techniques such as Magic 123 and Wonder3D will be thoroughly analyzed.

3.2.1 Magic 123

"**One** Image **to** High-Quality **3D** Object generation using both 2D and 3D diffusion priors" [Qian et al., 2023] - Magic123 - generates 3D meshes using a coarse-to-fine manner as already seen in other methods. The approach is distinctively characterized by its ability to balance between imaginative exploration using 2D priors and precise exploitation of the generated geometry with 3D priors [Qian et al., 2023].

At its core, Magic123 begins with the segmentation of the object from the background using the Dense Prediction Transformer model [Ranftl et al., 2021]. This step extracts the foreground object with its mask, which is then processed further with the MiDaS [Ranftl et al., 2020] model to generate a depth map, preventing the flattening of geometry and capturing the object's actual geometric details [Qian et al., 2023].

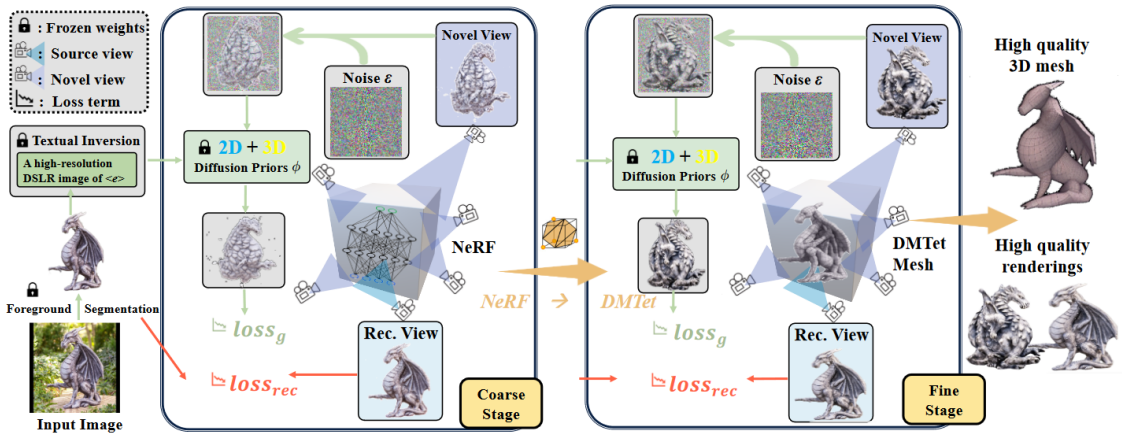


Figure 13: Overview of Magic123's two-stage process, which starts with the initial image and illustrates the transition from coarse geometry capture with Instant-NGP to high-resolution mesh refinement using DMTet. Image taken from [Qian et al., 2023].

In the coarse stage, Magic123 focuses on capturing the underlying geometry that from the extracted object. It uses Instant-NGP as the Neural Radiance Field (NeRF) implementation due to its enhanced memory efficiency. The training approach in this initial phase involves a sequential process where the model first samples from the Instant NGP representation. Following this, the model undergoes optimization using multiple loss functions. The Reference View Reconstruction Loss, $\mathbf{L}_{rec}$, ensures that the image rendered by the NeRF from a predetermined viewpoint closely aligns with the reference image from the diffusion model Stable Diffusion [Rombach et al., 2022]. Alongside, the Novel View Guidance Loss, $\mathbf{L}_g$, is employed. This loss function uses both 2D and 3D diffusion priors to aid in creating novel views of the object. It guides the training process towards more accurate results [Qian et al., 2023]. Additionally, depth prior and normal smoothness are implemented to overcome inherent challenges in reconstructing 3D content from 2D images, such as avoiding flat or caved-in geometries and smoothing high-frequency artifacts on object surfaces. Despite these advanced techniques, the coarse stage faces a limitation in terms of resolution. Even with Instant-NGP, the maximum resolution achievable in the image-to-3D pipeline is capped at 128×128 on a 16GB memory GPU [Qian et al., 2023].

The fine stage aims to refine the 3D model into a high-resolution mesh using Deep

Marching Tetrahedra (DMTet) [Qian et al., 2023; Shen et al., 2021]. This stage effectively disentangles geometry and texture, overcoming the resolution limitations of the coarse stage and enabling the generation of meshes at a much higher resolution. The fine stage mirrors the coarse stage in structure but is distinguished by its use of DMTet for the 3D representation and loss evaluation using the viewpoint-conditioned diffusion model Zero-1-to-3 [Liu et al., 2023], where both significantly enhance the level of detail and realism in the final model [Qian et al., 2023].

Both stages of Magic123 use both 2D and 3D priors to guide the generation of novel views. The 2D priors, powered by Stable Diffusion, excel in imaginative extrapolation and contribute to exploring the space of possible geometries. However, they are constrained by their limited understanding of three-dimensional structures, leading to issues like unrealistic geometries or inconsistencies in textures [Qian et al., 2023]. Contrastingly, the used 3D priors generated by viewpoint-conditioned Zero-1-to-3, provide a more accurate and consistent representation of three-dimensional objects. These priors excel in maintaining geometric fidelity but can struggle with generalizing to less common objects, sometimes resulting in oversimplified geometries. [Qian et al., 2023].

3.2.2 Wonder 3D

Long et al. introduce Wonder3D, a technique that stands apart from its predecessors primarily through its adeptness in generating color images and consistent multi-view normal maps. This method leverages a cross-domain diffusion model, which "extends the stable diffusion framework to model the joint distribution of two different domains, i.e., normals and colors" [Long et al., 2023].

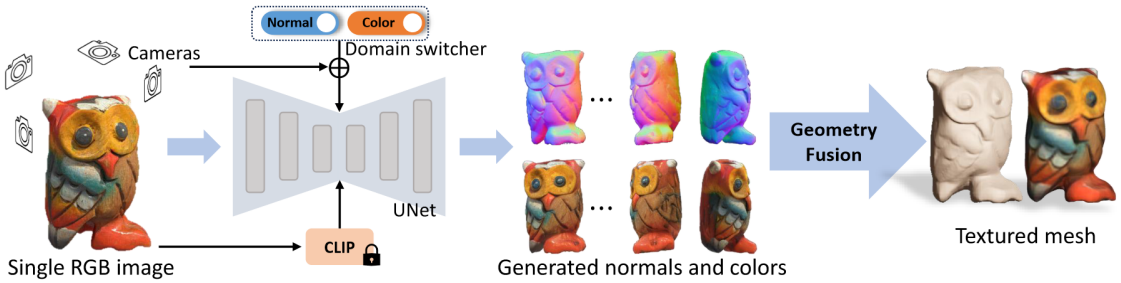


Figure 14: This Figure adapted from Long et al. [Long et al., 2023] summarizes the functionality of Wonder3D, illustrating its unique approach in generating textured meshes from single images using cross-domain diffusion models.

The process of generating 3D models from images begins with the use of CLIP [Radford et al., 2021] to obtain a textual description of the image. Wonder3D allows for training on 2D diffusion models by using a unique formulation of 3D models. The distribution of 3D assets is provided "as a joint distribution of its corresponding 2D multi-view normal maps and color images" [Long et al., 2023].

Given an image y, the model f aims to produce multiple normal maps $n^{1:K}$ and color images $x^{1:K}$ from various camera positions $(\pi_1, \pi_2, \ldots, \pi_k)$, mathematically represented by Long et al. as $(n^{1:K}, x^{1:K}|y) = f(y, \pi_{1:k})$.

The creation of the cross-domain joint distribution in Wonder3D is enabled by the use

of a Markov chain as part of a diffusion model. Long et al. formally represent this as

$$p\left(n_T^{(1:K)}, x_T^{(1:K)}\right) \prod_t p_\theta \left(n_{t-1}^{(1:K)}, x_{t-1}^{(1:K)} \mid n_t^{(1:K)}, x_t^{(1:K)}\right)$$

where $p\left(n_T^{(1:K)}, x_T^{(1:K)}\right)$ is the Gaussian noise [Long et al., 2023]. For each step t, the model refines the normal maps and color images from the previous step $t-1$ by refining the model's parameters p_θ.

Simply adapting pre-trained stable diffusion models to also output both normal maps and color images, encountered issues with slow convergence and poor generalization [Long et al., 2023]. To address these challenges, Wonder3D introduce a cross-domain diffusion scheme named Domain Switcher s, which is given as an extra input to the diffusion model [Long et al., 2023].

$$n^{1:K}, x^{1:K} = f(y, \pi_{1:K}, s_n), f(y, \pi_{1:K}, s_c)$$

This equation by Long et al. means that the model uses the switcher s to decide whether to generate normal maps (s_n) or color images (s_c), based on the input provided. To ensure geometric consistency between the color image and the normal map for a single view, Wonder3D employs cross-domain attention, allowing for information exchange between multiple domains [Long et al., 2023].

The transformation from 2D outputs (normal maps and color images) to 3D geometry is achieved by optimizing a neural implicit signed distance field (SDF). This optimization process is geometric-aware, meaning it takes into account the spatial relationships and shapes present in the 2D inputs. This involves segmenting object masks from normal maps or color images, and optimizing by "randomly sampling a batch of pixels and their corresponding rays in world space [...]" [Long et al., 2023]. This method of sampling and adjusting points in the SDF based on the rays ensures that the 3D model is a true representation of the object as seen in the 2D images.

Long et al. define the overall objective function as:

$$L = L_{normal} + L_{rgb} + L_{mask} + R_{eik} + R_{sparse} + R_{smooth}$$

Each term serves a specific purpose: L_{normal} alligns the 3D geometry with the generated normal maps, by employing a cosine function to maximize the similarity between the normals of the signed distance field (SDF) and the generated normals [Long et al., 2023]. L_{rgb} ensures color accuracy, and L_{mask} calculates the errors between masks [Long et al., 2023]. The eikonal regularization term R_{eik} maintains the stability of the shape, the sparsity regularization term R_{sparse} avoids isolated and floating parts in the SDF and the smoothness regularization term R_{smooth} contributes to the natural appearance of the 3D geometry [Long et al., 2023]. It is important to note that at the time of writing this thesis, the sparsity regularization term is referred to as L_{sparse} and the smoothness regularization term as L_{smooth} in the original paper, which could indicate a simple spelling mistake as they are given as R_{sparse} and R_{smooth} in equation (6) [Long et al., 2023].

Despite its innovation, Wonder3D has some limitations, as the method can only generate normal and color images from a limited number of six views, which hinders the ability to create intrinsic and finely detailed objects [Long et al., 2023]. While increasing the number of views could potentially enhance the detail and quality of the output, it would also significantly escalate computational demands [Long et al., 2023].

Chapter 4

Comparative Study

This chapter deals with a systematic evaluation of the previously investigated methods for creating 3D models. This analysis is crucial for understanding the real-world applicability and effectiveness of each method. The aim is to compare the theoretical principles with the practical results and to provide a comprehensive evaluation of the performance of each model under experimental conditions.

The chapter begins with a description of the experimental setup that was created to test these models. This includes the specific conditions, frameworks and parameters used to ensure that the comparison is fair and the results reliable.

The generation process of each model is then presented, showing how each model is derived from its inputs into a 3D model. This also includes measures of performance, including generation time, resource efficiency and versatility of each model, providing a multi-dimensional overview of the capabilities of each method.

Finally, the chapter highlights a thorough analysis of the results obtained from these experiments. This section not only presents the data, but also interprets them in the context of the theoretical background, limitations and potential applications of each model. The insights gained here are helpful for anyone who wants to understand the state of the art in 3D modeling and its practical implications in the real world.

4.1 Experimental Setup

3D model generation is a demanding task, both in terms of computational power and hardware resources. Each method has unique requirements, which poses a significant challenge in creating a fair baseline for comparative analysis.

To address this, the GitHub project Threestudio [Guo et al., 2023] was utilized. This platform provides slightly adapted versions of the official methods, preserving their core functionality while making them more accessible for systems with limited hardware capabilities. Magic123 [Qian et al., 2023], for example, was originally tested on a V100 GPU with about 32GB RAM, but Threestudio's version also works on a T4 GPU with about 15GB RAM. This adjustment ensures uniform testing conditions across various methods, enabling fair comparisons without the need for high-end hardware. However, this benefit comes at the cost of extended training times and inferior outputs compared to the original versions. Threestudio's modifications can be found in the Appendix Chapter B.

Despite the advantages of Threestudio, hardware limitations were still a major problem. The available hardware for this thesis was a single NVIDIA GeForce RTX 2080 GPU with 8GB of RAM. Therefore, Google Colab [Google LLC, 2023] was used to mitigate these restrictions as it provides access to free GPUs (with about 15GB RAM) and the ability to run code efficiently. However, this approach had its own limitations. Colab is limited to a single GPU, whereas most 3D generation methods typically benefit from training with multiple GPUs, to achieve more detailed results and faster computation times. Another notable drawback of Google Colab is its unpredictability in terms of how long a notebook can be used before the runtime is reset. Opting for the premium version of Colab can mitigate this issue by providing a more stable runtime, but this comes with a subscription fee.

Threestudio provided a test Colab notebook with an initial implementation for DreamFusion. This setup was further refined and extended by myself to improve its functionality and usability. Changes included the ability to transfer the complete training folder with checkpoints, validation images, configuration details and outputs to Google Drive for easy data access and storage. In addition, this improved notebook now includes comprehensive code snippets for training, refining and exporting for each model beyond the scope of Dreamfusion. To prevent common computational errors due to varying dependencies, additional packages were integrated into the setup. In order to combine all methods used in this thesis in one notebook, the official implementation of Wonder3D [Long et al., 2023] is also included, as well as Evaluate3D, a tool I developed myself for basic geometric comparisons of object files.

It is worth mentioning that the functionality of this notebook is strongly based on Threestudio's and Wonder3D'S guidelines, and as the project evolves, some personal adjustments might become unnecessary due to updates by the original authors.

4.2 Individual Generation Process

If not mentioned differently, each model was trained for 10,000 iterations using a single T4 GPU in Google Colab with the High-Ram setting enabled. It is important to note that this hardware configuration does not match the specifications used in the official implementations of these models. Consequently, the results generated in this study may not be as detailed and precise. Despite these limitations, the primary purpose here is to evaluate the basic capabilities of each model, which is possible even with comparatively modest hardware standards.

To effectively demonstrate the generation process of each method, an example prompt was used: "a robot made of plants". The choice of this prompt was strategic as concept of a robot is inherently versatile and does not have a rigid definition in terms of appearance or composition. Furthermore, it was assumed that a simple prompt such as "a robot" would lead to bland and colorless models, reflecting the results observed when applying such a prompt to a text-to-image model, specifically Dall-E 3 [Ramesh et al., 2023; Betker et al., 2023]. To mitigate this, the phrase "made out of plants" was added. This not only countered the potential monotony of the models, but also tested the models' ability to represent intricate detail and incorporate color, particularly the various hues associated with plants. The expectation was that this addition would enrich the output of the models and provide a more comprehensive basis for evaluating their detail rendering capabilities.

Dreamfusion – This method initiates the modeling process with a random scene, which it then incrementally refines throughout the training period. A unique aspect of this method is that each prompt triggers the formation of a new scene, ensuring that even when the same textual input is used repeatedly, it results in the creation of distinct objects. This feature of Dreamfusion is showcased through the results visible in Figure 15 and Figure 32, both of which are included in the Appendix. For the purposes of this research, the stable-DreamFusion variant [Tang, 2022], as available in Threestudio, was utilized. This version is distinct from the original Dreamfusion implementation, primarily due to the unavailability of Imagen to the public. Details regarding any other modifications made in this version, compared to the original, are outlined in the Appendix.

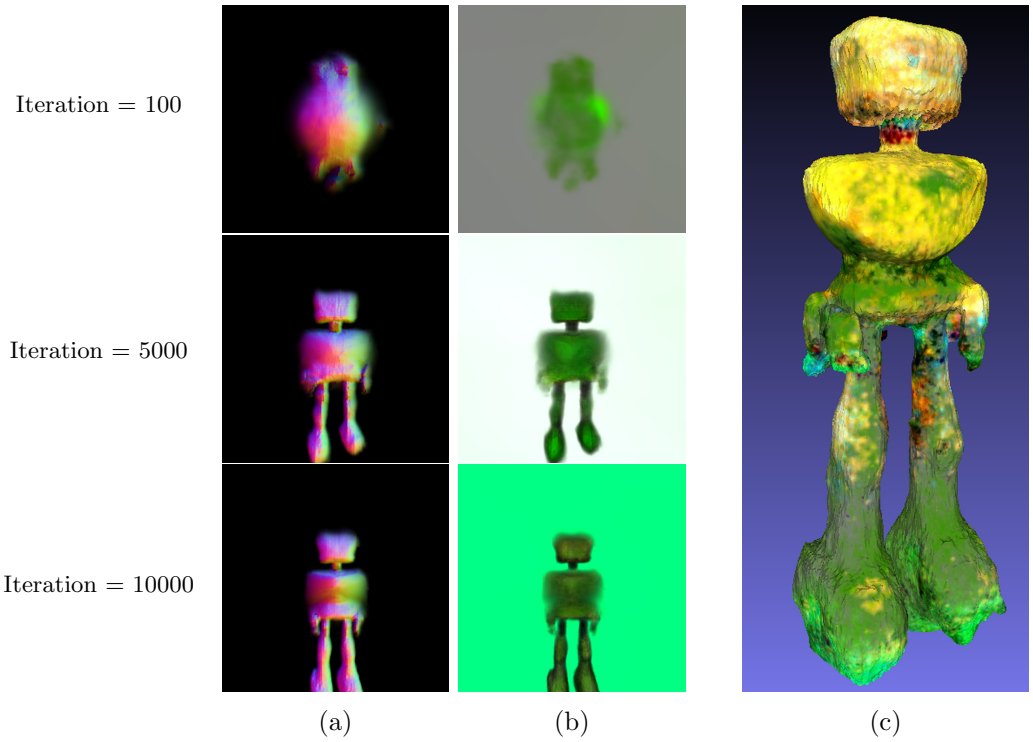


Figure 15: The generation process of Dreamfusion using the prompt "a robot made out of plants". Section (c) shows a snapshot of the final mesh generated.

As seen in Figure 15 parts (a) and (b), the object and its texture are generated simultaneously. The process starts with a small dot that gradually transforms into a more complex shape. At the 100th iteration, the original dot begins to transform into a recognizable shape, and a green hue resembling plant coloration is created. At the 5000th iteration, distinct features such as two legs, a square body and a square head become visible, all retaining the same shade of green. At the last, 10,000th iteration, the model shows a background and small arms sticking out of the robot's body. Part c of the figure showcases the rendered mesh opened in Meshlab [Cignoni et al., 2008]. During the mesh conversion phase, Threestudio made some changes, such as removing duplicates and filling holes. These changes are the reason for the slight differences between the final mesh and the validation images created during training. Interestingly, the final mesh lacks detailed

plant-like features, one could only assume that the legs kind of resemble moss, but this remains speculation. The mesh primarily shows basic shapes, including a square head, a torso with small protruding rods that could be arms, and large legs. Remarkably, the upper half of the body in the final mesh takes on a yellowish color that differs from the green of the earlier validation images. The reason for this color change remains unclear.
Magic3D – Magic3D employs a coarse-to-fine methodology, aiming to first construct a basic outline of the target object, which is then refined in subsequent stages to more accurately align with the text prompt. The mesh initialization is random, mirroring the approach used in DreamFusion.

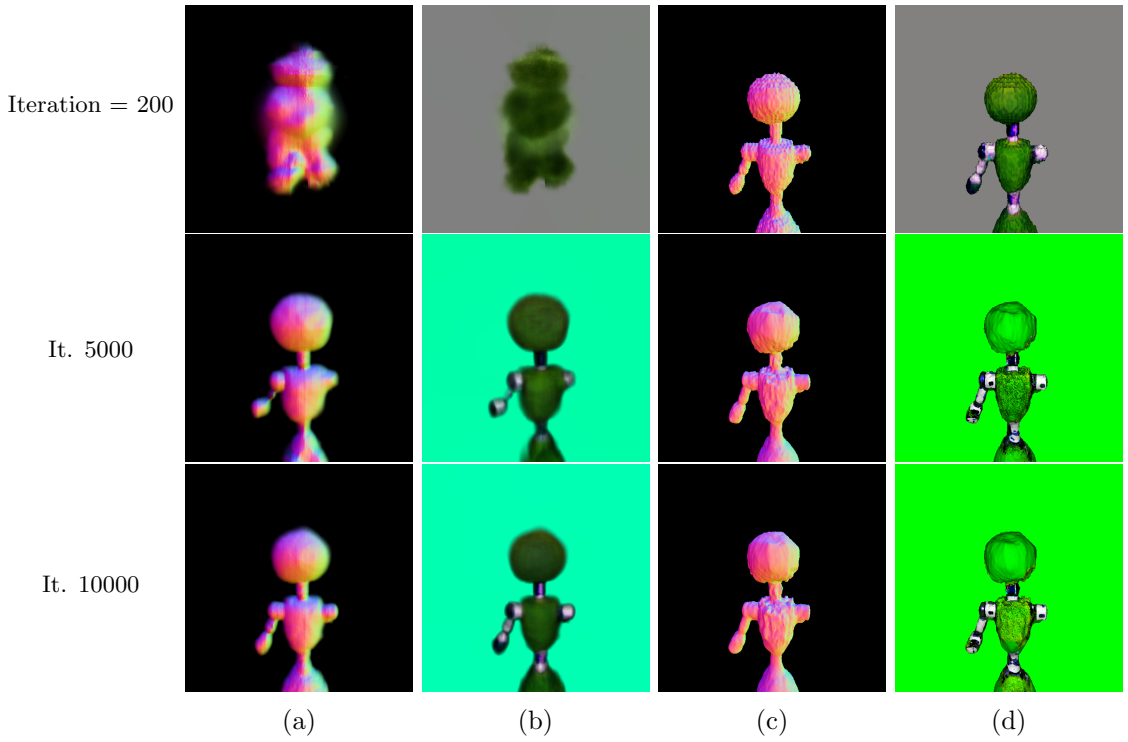


Figure 16: Magic3D generation process from coarse (a, b) to fine (c, d)

The coarse stage can be seen in Figure 16 parts (a) and (b). From a random start, a rough shape starts emerging by Iteration 200, accompanied by a plant-like green hue, setting the foundation for further refinement. By Iteration 5000, significant transformations have occurred from the initial stage: a distinct circular head, a neck, the upper body of the robot, and one arm have formed. The lower body, however, was omitted during training, forcing subsequent refinements on the upper half. At this point, the arm's color diverges from the green base, taking on a rough, grey-metallic appearance. Progressing to Iteration 10000, the neck and certain parts of the hip also adopt this metallic color. Despite these changes, the overall shape of the model sees only minor adjustments between Iteration 5000 and 10000, such as a thicker left arm and a more pronounced hip, but the right arm remains entirely absent. From the coarse stage alone, the model vaguely indicates a robotic form, with the plant aspect being derived primarily from the green coloring.

In the refinement stage, parts (c) and (d), the process starts from the model generated

in the coarse stage. Initially, the mesh appears blocky but retains its original shape, with the neck, shoulder, and hip areas acquiring a purple hue between Iterations 0 and 200. By Iteration 5000, the model is smoother, with minor modifications to the head, losing some of its roundness. However, not much else changes in shape. The texture, though, sees significant refinement; the arms, hip, and neck gain more detail, resembling parts of an actual robot. The chest acquires grass-like detail and coloration. By Iteration 10000, some of these textural details diminish, as evidenced by the stomach area reverting to a plain green. However, the model gains light reflections, particularly noticeable on the shoulders. Despite these changes, the model's shape remains largely unchanged from Iteration 5000 to 10000, and the missing arm issue persists through the refinement stage.

The final mesh, as displayed in Figure 17, is recognizable as a robot, and with close inspection, one might discern its grass-like chest, suggesting a plant-themed robot. For immediate and clear identification, the model would benefit from enhanced detail, particularly a more developed lower body extending beyond the hips. The left side of the figure displays the albedo generated during training.

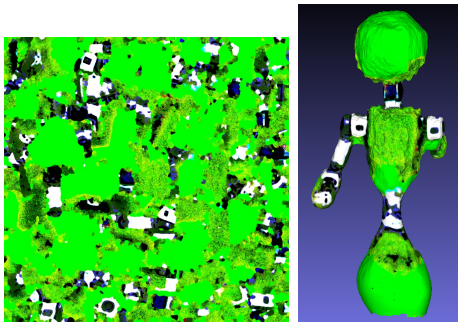


Figure 17: Magic3D also generates an albedo during training. The right side shows the extracted mesh.

Fantasia3D – There are several ways to initiate the generation of 3D models in Fantasia3D. The most straightforward method, used in this section, is to begin with just the prompt, wherein Fantasia3D defaults to using a sphere as the initial mesh, shaping the training process from this starting point. Alternatively, one can customize the sphere's initial values $[0.5, 0.5, 0.5]$ to better represent the desired object by adjusting the parameters to correspond to $[depth, width, height]$. The final approach involves initializing the mesh with a custom .obj file, providing a rough outline of the intended shape. An example of the latter approach can be seen in Figure 33 in the appendix, where the generation process was started with a rough human figure.

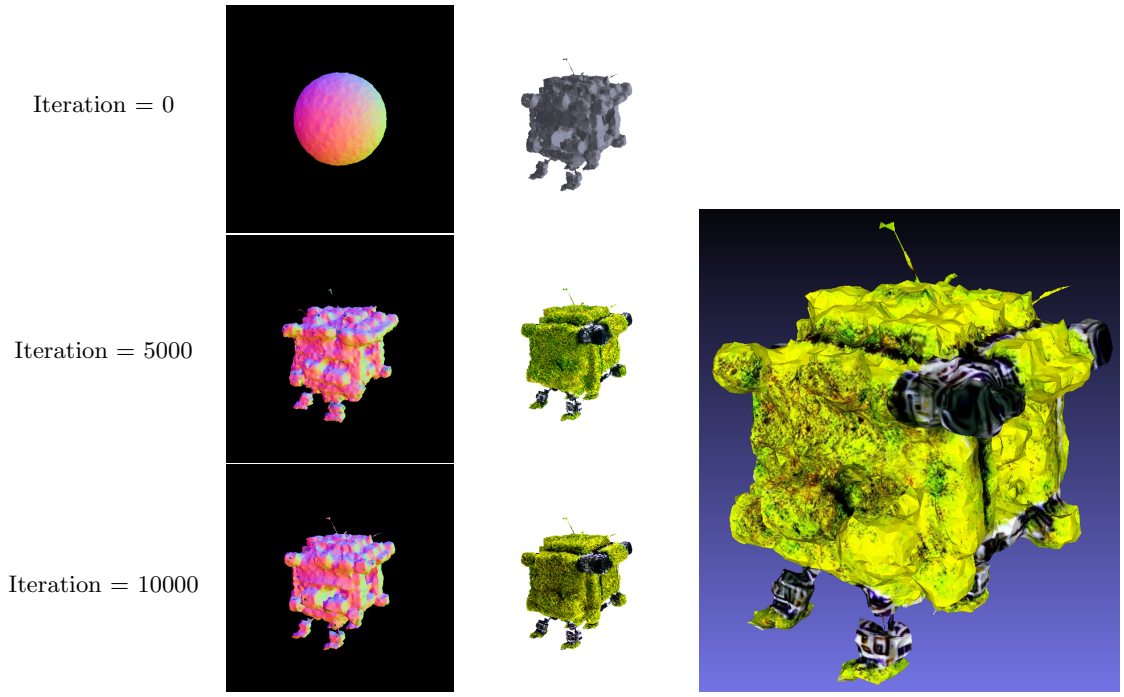


Figure 18: a: Fantasia3D starting with only a perfect square and refining this according to the prompt "a robot made out of plants". In b: only the appearance of the model gets refined. Image c shows the rendered model

Part (a) of Figure 18 illustrates the geometry stage of the method. Since only the prompt was used for initialization, Fantasia3D automatically selected a sphere as the base, influencing the overall quadratic shape of the final model. The figure reveals that the majority of the transformations occur between iterations 0 and 5000, where the sphere evolves, gaining corners and forming smaller blobs at the bottom, potentially interpreted as the robot's feet. However, at this geometry stage, it's challenging to discern the robot, especially as one made of plants. The changes from iteration 5000 to 10000 are more subtle, with slight smoothing in certain areas, such as the blob on the top left side of the robot or parts of the left foot, but these alterations are not significantly transformative.

Part (b) of the figure displays the appearance stage, where the model is textured. Starting with a greyscale base derived from the previous stage, the model gains color and texture as iterations progress. By iteration 5000, the texture, surprisingly resembling grass, enhances the model's detail and color, diminishing the prominence of the earlier clunky geometry. From iteration 5000 to 10000, this texture is further refined, introducing more detailed color variations and shadows, simulating light effects. Additionally, certain areas of the model exhibit metallic grey tones, particularly noticeable at the top right side and between the main body and the feet.

However, some of these textural details are lost in the final mesh extraction, as seen in part (c). The model takes on a more yellowish hue, and the previously distinct lighting and shadows are reduced. Similar to the geometry stage, the final model does not clearly represent a robot made of plants; it more closely resembles a box with feet, akin to robots

designed for food delivery. Throughout the training, Fantasia3D also generates various textures, including diffuse, roughness, metallic, and normals, as depicted in Figure 19.

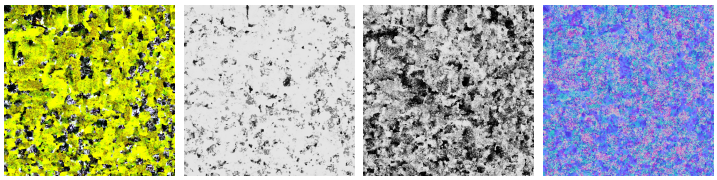


Figure 19: Generated textures from Fantasia3D; from left to right: diffuse, roughness, metallic, and normal.

Magic123 – Magic123 employs a unique approach that integrates both 2D and 3D priors for generating 3D objects. The method initiates by processing an input image, creating multiple perspectives of the intended object. The input image, depicted in Figure 22 part (b), was generated using the same prompt with Dall-E 3. The initial phase involves constructing a low-quality, coarse model complete with texture, forming a basic representation of the object. Subsequently, this model undergoes a refinement stage, where additional details are added to both the object and its texture.

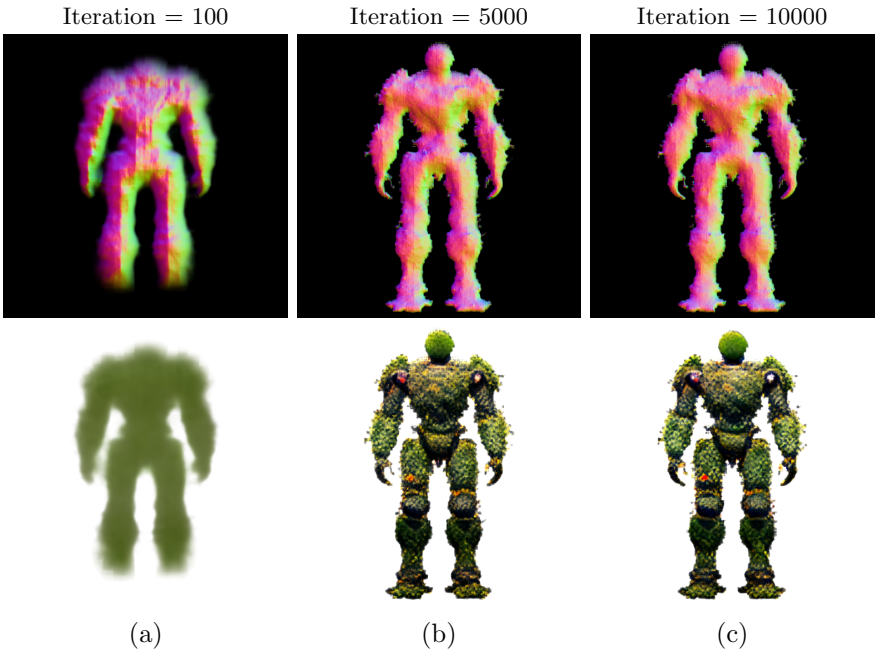


Figure 20: Front view of the coarse stage of Magic123

The front view of the coarse generation process is depicted in Figure 20. Even at an early stage like Iteration 100, the method efficiently produces an outline that resonates well with the input image, including the incorporation of a green hue to represent the plant aspects. Progressing to Iteration 5000, the model becomes more detailed, with sharper edges and a more defined robotic shape. Elements suggestive of overgrowth, such

as leaves and vines, begin to emerge, and the texture convincingly integrates plant-like features, covering the model with grass and vine patterns. By Iteration 10000 in this stage, the model's shape does not evolve significantly, but some color variation in the plant textures and distinct features like a white spot on the right shoulder and a bright red dot above the left knee emerge, hinting at metallic robot parts and a blooming flower, respectively.

During the refinement stage, showcased in Figure 21, the model initially loses some of its detailed and mossy character from the coarse stage. Floating parts disconnected from the main body become noticeable, particularly on the right arm and the left leg. Compared to the 10000th iteration of the coarse stage, the texture seems less detailed initially. However, by Iteration 5000, the texture regains and even surpasses its former level of detail, presenting a more realistic plant-like structure. The model appears covered in grass, moss, and vines, with enhanced features like a more pronounced blooming flower and more defined hands, knees, shoulders, and feet. Further into Iteration 10000, the model becomes smoother, particularly around the thighs and chest, though the floating parts persist. The texture at this stage is remarkably detailed, offering a realistic plant appearance with effective light and shadow interplay, particularly around the stomach area.

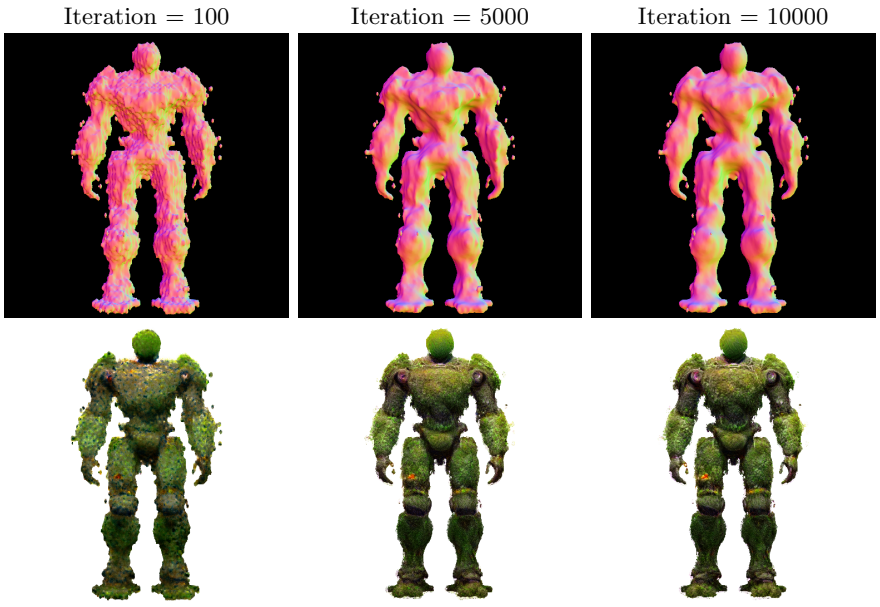


Figure 21: Front view of the refine stage of Magic123

As Magic123 generates multiple views during training, additional perspectives of the object for each iteration can be seen in Figures 34 and 35 in the appendix, which include the right, back, and left views. In there one can see the difficulties magic123 had in the beginning while determining the side views of the model. These problems however quickly dissapeared until Iteration 5000.

The final mesh, as illustrated in part (a) of Figure 22, exhibits reasonable quality. Some white spots, particularly around the floating parts, suggest that these areas may remain

textureless due to rendering issues. In comparison, the final model does a commendable job in mirroring the overall shape of the input image. Despite this, it becomes evident that Magic123 struggles to replicate the complex details of the plants. This shortcoming highlights the limitations of the method when it comes to fully capturing and translating the details in the input image into the 3D model. However, longer training iterations and higher computing power could mitigate this.

(a) Magic123 (b) Input Image (c) Wonder3D

Figure 22: 3D models generated by Magic123 and Wonder3D based on an imput image

Wonder3D – This method operates by leveraging a Domain Switcher technique when provided with an image. It produces color images and normal maps from multiple viewpoints of the given image, ensuring consistency across these views. This approach is pivotal for obtaining a comprehensive understanding of the model, not just from the front perspective but from all angles. In this particular test, the same input image as used for Magic123 was employed, which can be viewed in Figure 22 part (b). Figure 36 in the appendix illustrates the six viewpoints generated by Wonder3D, including front, front-right, right, back, left, and front-left views, each accompanied by its respective normal maps and color images. Utilizing these outputs, the model strives to construct a coherent and consistent 3D representation.

Contrasting with other methods, Wonder3D does not currently provide detailed validation images throughout the training process. Instead, outputs are available only at intervals of every 3000 iterations, as depicted in Figure 23. Without the ability to identify the relevant part of the original code to change this interval, progress can only be observed through these less frequent updates. From the available images, it is observed that the improvements in the model, based on the initially generated normals and color images, are relatively minor over the course of 10000 iterations. These slight enhancements are manifested in modest improvements in detail and the overall shape of the model, but they do not represent significant advancements from the initial stages.

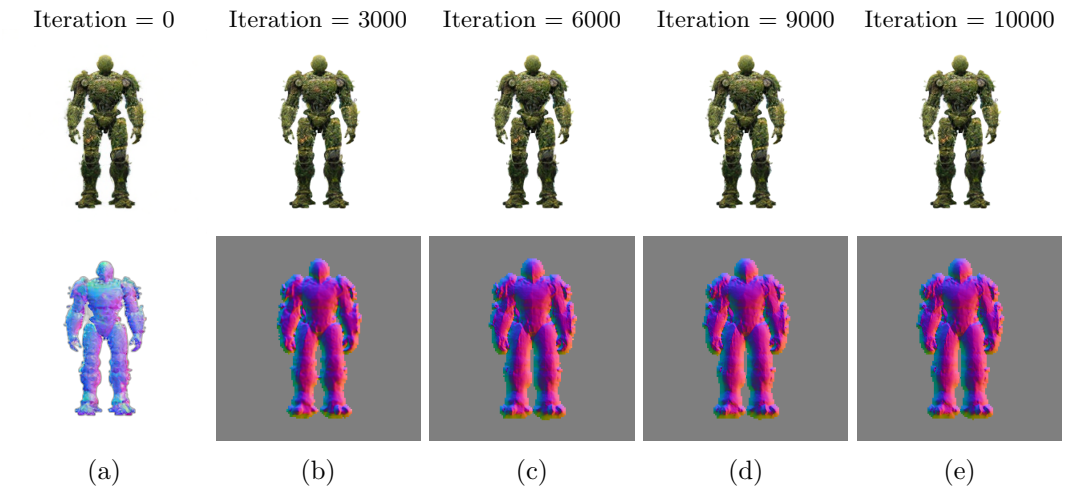


Figure 23: Front view of the Wonder3D generation process. Only minor changes can be seen between initialization and iteration 10000.

4.3 Comparative Analysis

Conducting a comparative analysis of 3D models presents a challenging endeavor, as the parameters defining their success vary widely based on the intended application. For instance, low-resolution models suffice for real-time rendering in virtual reality or gaming contexts, whereas the film industry demands high-quality renderings. Similarly, in industrial applications, precision down to the millimeter is crucial, necessitating extremely detailed and accurate meshes. This analysis therefore approaches the assessment of generated models from multiple dimensions:

1. **Prompt/Result Fidelity:** This dimension explores the alignment of each model with the original prompt and/or image. It questions whether the intended subject of the model is recognizable without prior knowledge of the input, thereby assessing the model's fidelity to the intended design.

2. **Geometry:** Here, the focus is on the complexity and intricacy of each model. It evaluates whether the models exhibit detailed, fine structures or lean towards a more generalized, simplistic design.

3. **Texture Realism:** This aspect assesses the authenticity and quality of the textures applied to the models. It delves into questions of realism - how real do the textures appear?

Following this subjective analysis, the study proceeds to evaluate technical aspects of each model, focusing on parameters such as rendering time, efficiency, and resource utilization. Each method is also subjected to targeted tests tailored to specific criteria. For instance, in scenarios where symmetry is critical, the models are evaluated for their ability to replicate symmetrical designs. Technical metrics for this analysis are derived from tensorboard data or outputs generated during the model training process. Detailed assessments and quantitative analyses - such as the evaluation of symmetry, and the

detection of holes or anomalies in the models - are examined using Evaluate3D. This custom-developed tool integrates the trimesh Python library [Haggerty et al., 2019] to provide insights into the geometric properties of each 3D model.

4.3.1 Subjective Evaluation

In the preceding section, it was observed that the methods yielded diverse outcomes when tasked with a broad prompt, such as creating a 'robot made of plants'. The text-to-3D methods faced challenges in initially generating an object, a stark contrast to the image-to-3D methods that benefited from having a reference image, offering some directional guidance and hence a slight advantage. To establish a more leveled playing field for the various methods, the next prompt chosen for testing was **"a high-quality rendering of a Playmobil firefighter"**. Playmobil figures are known for their uniform base structures, differing primarily in clothing or texture. This prompt was therefore selected to assess whether a method could accurately capture the fundamental structure dictated by the prompt. The outcomes of each method, applied to this specific prompt, are illustrated in Figure 24.

(a) DreamFusion (b) Magic3D (c) Fantasta3D (d) Magic123 (e) Wonder3D

Figure 24: Results obtained using the prompt "a high-quality rendering of a Playmobil firefighter".

Prompt/Result Fidelity: There is a clear split in the results when looking at fidelity to the prompt. DreamFusion and Magic123 tend to stick to the red clothing, while Fantasta3D, Magic123 and Wonder3D opt for a more realistic yellow and black firefighter uniform. Wonder3D and Magic123 adapt their results to the expected color scheme of the input image, given in Figure 25 part (a). Of particular interest is Fantasta3D's deviation from the red color scheme, although all models were refined with Stable Diffusion. This can be attributed to the physically based material model in the appearance stage, which has learned to associate firefighter clothing with yellow and black rather than red. Each model successfully reproduces identifiable Playmobil features such as helmets and uniforms, although the extent to which they reproduce the Playmobil style varies.

Geometry: The geometric shape of the models varies significantly. DreamFusion and Wonder3D present the most deviations from the expected Playmobil figure geometry. Both exhibit overly smooth shapes with ambiguous edges and floating elements, such as the hands in DreamFusion and around the right foot in Wonder3D, giving an unfinished appearance. Magic3D impresses with a near-perfect rendering of the Playmobil figure, capturing the sharp transitions at the joints and maintaining smoothness elsewhere, except for a slight distortion around the feet. This result seems like one could bend and move it like it is possible with an original figure. For this prompt, Fantasia3D was initialized with the model derived from Magic3D due to its solid performance, and not with a sphere as is typical. However, this Method still encountered issues with rendering unintended additional objects. These included a presumed fire extinguisher on the back and an ambiguous 'thing' at the front, possibly an attempt at rendering a fire hose. Magic123, while delivering a solid representation, introduces a large bag on the figure's back, which, unlike the extraneous parts of Fantasta3D, integrates well with the overall model. A closer look of this bag is given in Figure 25 parts (b) and (c). Uniquely, Magic123 recreates the iconic Playmobil hand structure, complete with thumb, fingers, and the characteristic gap between them, capturing the essence of the figure's appendages.

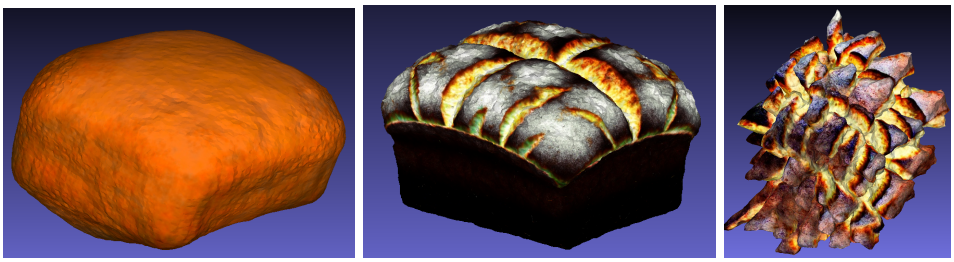
(a) (b) (c)

Figure 25: (a) displays the original image for the playmobil figure derived form Dall-E 3; (b) and (c) show the side and back view of Magic123, resectively.

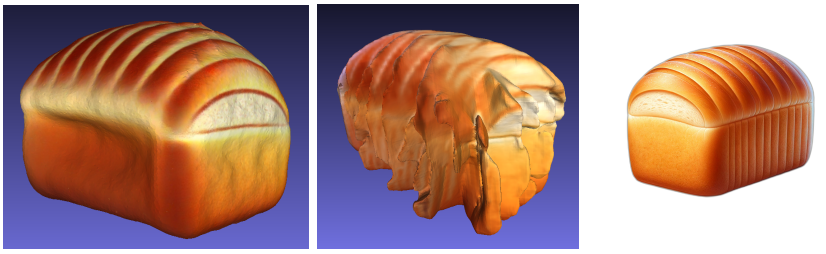
Texture Realism: In terms of texture, Magic3D again stands out with a level of realism that surpasses the others for the Playmobil prompt. It is characterized by a clear distinction between the colors, such as the black of the shoes against the red of the uniform, and the face retains the characteristic Playmobil look. The model's interaction with light, evidenced by chest reflections and inner leg shadows, adds to the plastic appearance. DreamFusion's model lacks such light interplay, resulting in a flat appearance, which is also caused due to the overall smoothened geometry. Fantasta3D, Magic123, and Wonder3D diverge from the red texture, opting for a black and yellow combination with realistic reflective stripes. Fantasta3D's texture is sound despite some geometric artifacts, with light reflection indicating a clear light source and giving the helmet a metallic sheen. Magic123 captures a plastic-like sheen, particularly on the arm's reflection, which underscores the roundness and smoothness of the figure. In contrast, Wonder3D's texture is not very detailed, blurs differences like the separation between jacket and shirt and does not render the character's face, similar to DreamFusion. The whole model looks strange, decayed and unfinished.

The results of this prompt show that DreamFusion struggles with detailed geometry, resulting in smooth and inaccurate shapes, but still offers acceptable color fidelity. Magic3D, on the other hand, excels in both realistic textures and geometric accuracy, especially when the prompt specifies a flat structure and no intricate detail is required. Fantasta3D tends to render overly realistic models, often at the expense of key prompt features, resulting in unnecessary additions in both geometry and appearance. Similarly, Magic123, while generally rendering the essence of the prompt well, tends to add unnecessary elements such as the unexpected bag. Finally, Wonder3D fails to produce even basic details, resulting in models that look unfinished and incomplete.

The next prompt used for the various methods was **"a rendering of a highly symmetrical loaf of bread"**. This prompt was selected to test the precision of each method in replicating detailed, specific requests while still producing coherent results. In this case, the model was tested for its symmetric result capability, which is discussed later in the technical section. The generated models are displayed in Figure 26, which also includes the original input image created by Dall-E 3 in part (f).



(a) DreamFusion (b) Magic3D (c) Fantasta3D



(d) Magic123 (e) Wonder3D (f) Original Image

Figure 26: Results obtained using the prompt "a rendering of a highly symmetrical loaf of bread". Part (f) is the input image for Magic123 and Wonder3D, generated with Dall-E 3

Regarding **Prompt/Result Fidelity**, the outcomes varied. Magic123 and Magic3D successfully captured the essence of a bread, while Wonder3D produced an image that only vaguely suggested a loaf of bread. DreamFusion and Fantasia3D, however, were less successful; the former's result was indistinct, resembling a nondescript lump, while the latter's output more closely resembled a pineapple or an explosion when viewed without context.

In terms of **Geometry**, DreamFusion's model lacked meaningful structure, while Magic3D's bread model achieved good geometry, with a realistically carved top resem-

bling fresh black bread. Fantasia3D, on the other hand, produced a model with a spikey appearance, potentially due to overfitting or difficulty in determining a starting point for the flat bottom of the bread. Magic123 excelled in replicating the geometry of the input image, closely matching the cuts and overall shape of the bread. Conversely, Wonder3D's model, like Fantasia3D's, had a roughly bread-like shape but suffered from spikey edges, possibly due to small adjustments during 3D mesh generation.

When considering **Texture Realism**, only the textures from Magic3D, Fantasia3D, and Magic123 seemed noteworthy. Magic3D's texture gave the impression of a burnt, dark loaf, with a realistic color gradient on the top. Although Fantasia3D's texture had a pleasing color combination, it resembled a cluster of breadsticks unless contextual information was provided. Magic123 accurately replicated the texture of the original image, demonstrating its proficiency in handling low-detail requirements.

To summarize, DreamFusion continues to have issues with structure, suggesting that direct prompts for certain features such as symmetry may lead to confusion rather than better results. Magic3D maintained its solid performance, suggesting that some level of directives in the prompt can be beneficial. Fantasia3D appeared to be thrown off by the specific prompt, resulting in poorer results compared to previous tasks. Magic123's exact replication of the input image leaves the impact of the specific prompt on its performance unclear. Wonder3D again struggled to accurately generate the overall structure, reflecting ongoing challenges in model training.

To evaluate the capability of various methods in rendering organic forms and textures, the prompt **"a high-quality rendering of a big dog sleeping on a chair"** was used. This prompt was specifically chosen to test the realism of fur texture and the anatomical accuracy of a resting dog. This setup also provided an opportunity to observe the effectiveness of multiview-consistency, a feature in Wonder3D, in enhancing the quality of the outputs.

For **Prompt/Result Fidelity**, most models, except Dreamfusion, produced a recognizable dog, as evidenced in Figure 27 and 28. However, all text-to-3D methods struggled with accurately rendering the chair's backrest, with Fantasia3D notably failing to make it clear what the dog was even resting on. Image-to-3D methods, Magic123 and Wonder3D, provided solid front views of the scene, though Magic123's representation from behind showed an unusually inflated and rounded chair shape.

In the aspect of **Geometry**, DreamFusion had difficulties rendering a recognizable dog shape, unlike the other methods. Magic3D's output was more successful, clearly differentiating between the dog and the chair from both front and back views, although an anomaly was noted where the dog appeared to have a total of four back legs in different views. This issue could stem from Magic3D's lack of multi-view consistency, in contrast to Wonder3D. Fantasia3D's model showed a dog, but with less detail and shape accuracy, especially in the front view where the legs seemed to emerge directly from the neck. Both Magic123 and Wonder3D did an impressive job of capturing the main features of the input image in the front view, as can be seen in part (f) of Figure 27. However, viewing from the back side, Magic123's rendering of the chair was less convincing, whereas Wonder3D's multi-view consistency resulted in a more accurate and flat back of the chair, despite some missing chair legs and floating parts.

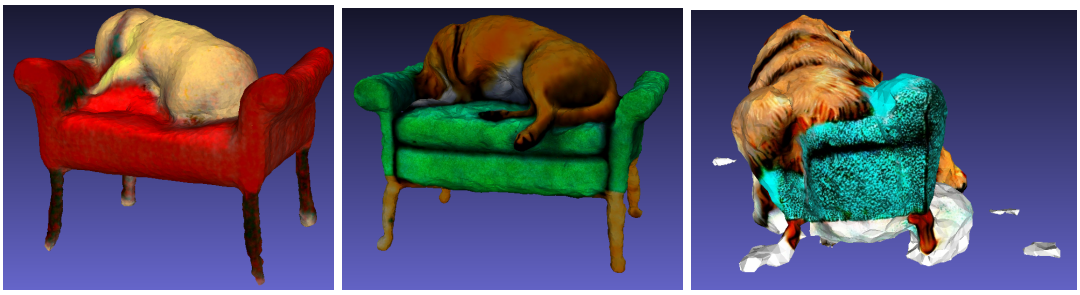
(a) DreamFusion (b) Magic3D (c) Fantasta3D

(d) Magic123 (e) Wonder3D (f) Original Image

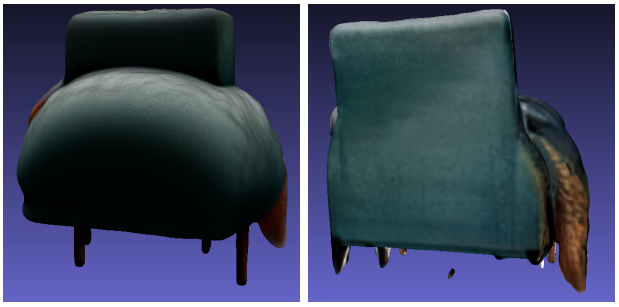
Figure 27: The front view of the generated objects with the prompt "a high-quality rendering of a big dog sleeping on a chair".

For **Texture Realism**, each method displayed distinct capabilities and limitations. DreamFusion's texture lacked explicit detail, resulting in a smoothed-out appearance that made it difficult to discern a dog lying on a chair. This lack of detail in texture diminished the overall realism of the scene. Magic3D, while not offering much depth in texture of the fur, managed to convey the impression of a dog through its effective use of color variations. The distinctions between the brown fur, black nose, paw gaps, shadows, and the white of the dog's neck and belly were clear, creating a convincing representation of a dog despite the geometric inaccuracies. Fantasia3D, although its geometric output was less accurate, excelled in texture detail. The color variations across the dog's face were more nuanced compared to Magic3D, adding a layer of realism. Even with the structural issues, the texture applied depth to the model and gave it a lifelike appearance of a dog. Magic123 and Wonder3D both achieved similar levels of detail in texture, closely mirroring the input image. Magic123, however, introduced additional lighting into the scene. This was evident in the reflections on the dog's back and face, enhancing the vitality of the overall scene. The texture of the chair's backrest was much finer Wonder3D and benefited from the interplay between shadow and light. This attention to detail made the fabric of the chair appear better.

DreamFusion again presented the lowest level of detail, failing to recognize the object specified in the prompt. Magic3D effectively captured the required elements, including the dog's resting anatomy, yet produced a flat, depthless texture for the fur. Fantasia3D accurately rendered parts of the prompt, focusing on the dog, striving for realistic texture but faltering in basic geometric accuracy. Magic123 performed well in replicating the



(a) DreamFusion (b) Magic3D (c) Fantasta3D



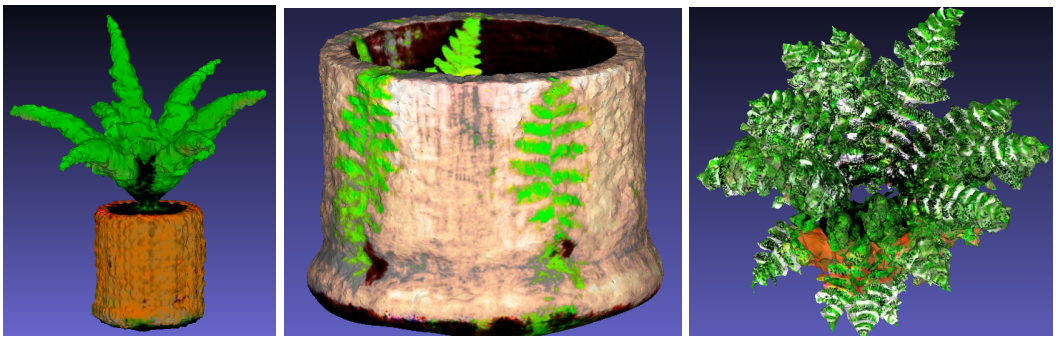
(d) Magic123 (e) Wonder3D

Figure 28: The back view of the prompt "a high-quality rendering of a big dog sleeping on a chair"

shape from the original image but struggled with rendering views different from the input. Wonder3D solidly recreated the input image, offering a consistent shape and texture from multiple viewpoints.

To delve deeper into each method's ability to render complex elements and contrasting materials, the prompt **"a high-quality rendering of a fern in a wooden pot"** was introduced. While plants had been used previously to illustrate each method's generation process, there hadn't been a comparison between the various Methods. This prompt aimed to evaluate how each method handled intricate details in natural subjects and the accuracy of color gradients and lighting in varying depths. The outcomes of this exercise are displayed in Figure 29.

In terms of **Prompt/Result Fidelity**, DreamFusion presented a fern and wooden pot that were recognizable, though it was debatable whether the plant could be distinctly identified as a fern. Magic3D struggled, failing to generate the fern and only displaying the pot with a texture vaguely resembling a fern. Interestingly, earlier in the coarse training, the fern was part of the model and was rendered, as seen in Figure 30, part (a). Fantasia3D showed a detailed fern but didn't render the wooden pot well. Magic123's front view effectively depicted both the fern and pot, but the side view lacked volume, making the plant appear flat compared to Wonder3D, which displayed a solid result for both the plant and pot from all angles, thanks to its multi-view consistency. The side views of Magic123 and Wonder3D are depicted in part (b) and (c) in Figure 30.



(a) DreamFusion (b) Magic3D (c) Fantasta3D

(d) Magic123 (e) Wonder3D (f) Original Image

Figure 29: Results obtained using the prompt "a high-quality rendering of a fern in a wooden pot".

Regarding **Geometry**, DreamFusion's model convincingly resembled a plant from all angles, with clear distinction between the pot and plant. Magic3D's pot was well-rendered with true shape and structure, but the plant went completely missing after the refinment stage. Fantasia3D exhibited detailed fern parts but failed to accurately portray the pot. Magic3D's model had a good pot shape, but the plant appeared flat and overly smoothed, possibly due to limitations in the input image, as white parts remained at the edges after the background was removed. Wonder3D presented a well-formed fern and pot from all views, although some floating parts and mixed-up sections were noticeable at a closer look.

For **Texture Realism**, DreamFusion's model allowed for basic identification of the pot as brown and the plant as green but lacked in shading and detail. Magic3D displayed three detailed fern leaves on the pot, but with incorrect positioning and lack of color variation due to the shape flaws. Fantasia3D achieved a realistic plant texture with significant depth and color variation, although the pot's color was less distinct. Magic123's texture appeared smooth with low detail in the plants, though the shadows and pot color matched well with the original image. Wonder3D showed detailed and varied colors within the plant, indicating depth, but the overall texture quality diminished upon closer inspection. However, the color consistency across all viewpoints and the well-rendered pot compared to the original image were notable strengths.

In summary, DreamFusion captured the basic shape and texture but lacked fine details, depth, and color variation, suggesting difficulty with intricate requests. Magic3D failed to accurately render detailed objects from the prompt, leading to an unsatisfactory output.

(a) Magic3D - coarse (b) Magic123 (c) Wonder3D

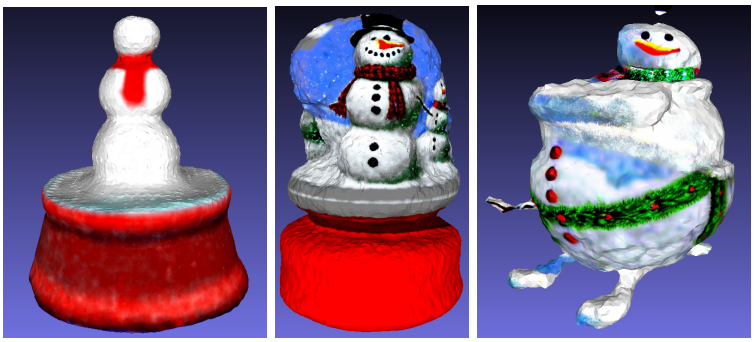
Figure 30: Part (a) shows the result of the coarse stage of Magic3D, where the actual fern was still present; (b and c) show the side view of the fern showcasing the limitations of Magic123 in deriving the correct angles in contrast to Wonder3D

Fantasia3D again achieved the most realistic texture but fell short in basic geometric aspects. Magic123 covered the basics, mirroring the view from the input image, but lacked detail, resulting in a flat, undetailed appearance. Wonder3D succeeded in capturing the basic form from all angles, with reasonable depth and structure. However, upon closer inspection, the texture appeared undetailed, and floating parts were noticeable.

The final prompt given to the methods was **"a detailed rendering of a snow globe containing a snowman"**, aimed at testing each method's ability to handle transparency and difficult reflections of glass. This prompt also evaluated the capacity to generate objects within other objects, like a snowman inside a glass globe. The results are depicted in Figure 31.

In terms of **Prompt/Result Fidelity**, no method could successfully render transparent glass with objects inside. This limitation might stem from the NeRFs' sampling process, where each view translates into a 2D image, potentially leading to a loss of three-dimensional structure information. Surprisingly, DreamFusion came closest to achieving the goal, capturing the intended object but failing to render the glass dome of the snow globe

Analyzing **Geometry**, there were significant difficulties across most methods. DreamFusion presented a well-structured podium and snowman, albeit lacking in detail and missing the glass dome. During training, this method depicted the dome with multiple floating parts representing one globe, as seen in Figure 32. However, these parts were removed in mesh conversion. Magic3D failed to produce a valid globe, resulting in an oval rather than a round shape. Fantasia3D seemed to focus more on the snowman, with an oversized belly potentially mistaken for the globe, and an inaccurately shaped snowman with legs and feet. Magic123's front view was promising, closely resembling the input image, but the side view, as seen in Figure 32, part (c), showed a thinner globe. Wonder3D stood out in its geometry rendering, presenting a circular dome from all viewpoints. Minor inconsistencies were observed at the bowl's connection to the podium, but these were minor compared to the other methods.



(a) DreamFusion (b) Magic3D (c) Fantasta3D

(d) Magic123 (e) Wonder3D (f) Original Image

Figure 31: Using the Prompt "a detailed rendering of a snow globe containing a snowman" to assess transparency and difficult reflections between various methods.

For **Texture Realism**, an interesting phenomenon was observed across all models but Wonder3D, known as the "Janus problem" as named by researchers like Ben Poole from DreamFusion. This issue involves multiple faces or elements being created in a single scene, leading to duplicated details from different viewpoints. For instance, DreamFusion had a scarf on both front and back sides, while Magic3D rendered five front-view snowmen within the globe. Fantasia3D's snowman had two faces and an additional snowman printed on its back. Magic123 also duplicated front-facing snowmen. Wonder3D, benefiting from its multi-view consistency, projected only a single snowman onto the globe. However, this approach made the snowman appear as if it was printed on the glass rather than inside the globe. DreamFusion's approach, which involved using floating particles to represent glass, lost these details during the mesh conversion process. Magic3D displayed ambiguous white spots on the globe, leaving it unclear whether they represented light reflections or snow. Although the snowmen models were detailed with accessories like scarves, hats, and buttons, this detail did not extend to effectively simulating the glass of the snow globe. Fantasia3D's snowman also had detailed elements but included multiple scarfs and a flat carrot nose. Magic123 attempted to replicate reflections from the original image, yet these efforts resulted in unrealistic outcomes. Similarly, Wonder3D also had a lack of transparency, as nothing inside the globe was discernible from the back, not even the snowman's texture visible from the front. The model attempted to add reflections, but

they fell short of the realism seen in the input image.

Overall, significant limitations were observed across all methods when rendering objects inside other objects, highlighting a current challenge in the field. Wonder3D, with its multi-view consistency, shows a promising approach, yet struggles like the others with achieving transparency. The interpretations of reflections from the input image led to errors, likely due to each method's unique approach to introduce lighting into the scene. This prompt resulted in the most ambiguous outcomes, highlighting major challenges and areas for future development in 3D model generation techniques

4.3.2 Technical Review

In the context of the thesis, each model was trained for 10,000 iteration steps, covering both the coarse and refine stages or geometry and appearance stages. Notably, the high RAM setting in Colab was enabled, primarily due to the substantial resource demands of Wonder3D. Even with this configuration, there were instances where the training process could be interrupted. Additionally, most of the settings were maintained as per the default Threestudio implementation, although there may have been some minor variations among the methods (some of these can bee seen in Appendix B). The reason for sticking to these default settings was primarily because altering them might have caused interruptions during training. Moreover, even with the preset configurations, the GPU RAM available on a T4 was often operating close to its maximum capacity.

Prompt	DreamFusion	Magic3D		Fantasia3D		Magic123		Wonder3D
		Coarse	Refine	Geom.	Appear.	C.	R.	
Robot	1:24	1:23	1:20	1:15	1:18	1:46	1:47	0:15
Playmobil	1:17	1:17	1:18	1:14	1:17	1:46	1:46	0:15
Fern	1:25	1:24	1:19	1:17	1:20	1:52	1:48	0:15
Bread	1:25	1:21	1:21	1:17	1:20	1:54	1:52	0:15

Table 1: Comparison of Generation Times for Different Prompts Across Methods (Hours:Minutes). Legend: C = Coarse, R = Refine, Geom = Geometry, Appear = Appearance.

The rendering time for each method was notably lengthy compared to 2D generative models, which can produce outputs in mere seconds, even with modest computational resources. Detailed information on the training times required for each method, along with various prompts, can be found in Table 1.

Among the methods, Wonder3D exhibited the most efficient performance, completing its training in approximately 15 minutes for 10,000 iterations. This is followed by Dreamfusion, which took about 1 hour and 15 minutes to 1 hour and 30 minutes. Magic3D and Fantasia3D demonstrated similar training times of around 1 hour and 15 minutes to 1 hour and 30 minutes each. However, it's worth noting that these two methods involve a two-staged process, resulting in a total training time of approximately 2.5 to 3 hours. The longest training duration was associated with Magic123, requiring approximately 1 hour and 45 minutes to 1 hour and 55 minutes for each coarse and refine stage, summing

up to around 3.5 to 3 hours and 50 minutes for 10,000 iteration steps. Considering these timeframes within the context of high-performance computing, which encompasses the use of superior and multiple GPUs operating simultaneously, leads to a notable reduction in training duration.

For reference, the setup used in the official paper is as follows:

- Dreamfusion: Utilized a single TPUv4, underwent 15,000 iterations, and completed training within 1.5 hours.
- Fantasia3D: Utilized 8 Nvidia RTX 3090 GPUs, with approximately 15 minutes needed for the geometry stage and 16 minutes for the appearance stage.
- Magic3D: Employed 8 Nvidia A100 GPUs, involving 5,000 iterations for the coarse stage with 1,024 samples per ray, taking 15 minutes, and 3,000 iterations for the refine stage, requiring 25 minutes.
- Magic123: Trained on a 32GB V100 GPU, with 5,000 iterations for both the coarse and fine stages. The coarse stage took around 40 minutes, while the refine stage needed approximately 20 minutes on a 32GB V100.
- Wonder3D: The pretraining phase encompassed the LVIS model, which included over 30,000 objects, and underwent 30,000 iterations on 8 Nvidia Tesla A800 GPUs, spanning approximately 3 days. Subsequently, the extraction of meshes from images could be accomplished in only 2 to 3 minutes.

The project Evaluate3D was then used in order to assess specific features of a given mesh, including the CLIP-Score and a rough symmetric value.

OpenAI's CLIP-score is a metric that quantifies the correspondence between an image and a given prompt [Radford et al., 2021]. This is achieved by encoding both the prompt and the image into high-dimensional vectors within the same embedding space, and then determining the cosine similarity between these vectors. Despite its sophisticated design, this score is not without limitations and at times cannot rival the discerning capabilities of the human eye, occasionally leading to outcomes that may seem counterintuitive.

For convenience and to streamline the evaluation process, portions of the code using this metric have been integrated into Evaluate3D. This enables an immediate calculation of the CLIP-score post-rendering. Utilizing this feature, scores for the Playmobil figures have been computed and are presented in Table 2. In an intriguing turn, Magic123 ranks highest when assessed against the original prompt, followed by Fantasia3D, Wonder3D, DreamFusion, and finally, Magic3D. These findings are not entirely in concordance with the subjective analysis provided earlier. However, when the prompt is changed to "a high-quality rendering of a red Playmobil firefighter", there is a marked reversal in scores. This suggests that the CLIP-score may exhibit a bias towards the color black in the context of firefighter apparel, as opposed to the more toy-like red. The discrepancies highlighted by the CLIP-score accentuate the need for the development of new, more precise evaluation metrics tailored for assessing the output of 3D generative AI, as there currently exists no standard method that is universally recognized as reliable.

Lastyl, the symmetry of some models is assessed. Evaluate3D uses a series of functions from trimesh to mirror a model along an axis and checks for corresponding vertices on the

Prompt	DreamFusion	Magic3D	Fantasia3D	Magic123	Wonder3D
a Playmobil firefighter	0.337	0.320	0.503	0.827	0.482
a red Playmobil firefighter	0.501	0.604	0	0	0.216

Table 2: CLIP-scores for Playmobil firefighter models based on different prompts.

mirrored side. This process is repeated for all vertices, and a symmetry score is derived based on the number of matching pairs found. The findings of this assessment are detailed in Table 3.

	DreamFusion	Magic3D	Fantasia3D	Magic123	Wonder3D
Symmetrie Score	0.337	0.320	0.503	0.827	0.482

Table 3: Symmetrie-scores for bread models demanding a symmetrical output.

Chapter 5

Future Directions

In this chapter, the focus shifts to the future of automatic 3D model generation, an area with a lot of potential but complex challenges. This section aims to bridge the gap between current methods and the visionary goals of the field. It addresses the pressing issue of generative bias, which is a critical problem as these models increasingly influence society's perceptions. In addition, the chapter explores emerging trends that are shaping the future landscape of 3D modeling. It examines how recent innovations are redefining efficiency and accessibility in model creation, with a particular focus on novel approaches such as Luma-AI's Genie [Jain et al., 2023] and Gaussian Splatting [Kerbl et al., 2023].

5.1 Emerging Trends and Future Directions

In the rapidly evolving landscape of text-to-3D model generation, the previously discussed methods represent only a fraction of the innovative approaches currently shaping the field. These methods, each with their unique features and methodologies, contribute to the broader spectrum of advances in 3D model generation. However, more recent developments, such as Luma-AI's Genie method [Jain et al., 2023], are pushing the boundaries further and overcoming many of the challenges faced by previous models.

Luma-AI's Genie, similar to the user-friendly approach of Midjourney [Midjourney, 2023], offers an accessible platform for 3D model generation. This service, operational on a Discord server [Discord Inc., 2023], allows users to input text descriptions. Upon receiving such input, Genie generates four preliminary models of the specified object. Users are then given the opportunity to refine these models. This iterative process, which typically takes around 10 minutes, culminates in the creation of a high-quality 3D representation. Notably, the use of Genie does not necessitate high-end hardware or additional platforms like Google Colab, as Luma AI manages the necessary computational aspects in the background. The model was published in November 2023 as a research preview to facilitate the creation of 3D models. However, no comprehensive details of how it works have yet been published.

The results achieved with Luma AI's Genie method are an example of the great progress made in this field. These advances demonstrate the potential of text-to-3D technologies for the fast and efficient creation of detailed, accurate 3D models.

Beyond 3D meshes or individual objects, the field of automatic 3D generation is also

expanding to the conversion of video to 3D, opening up a new dimension of realistic scene creation. The latest research in this area uses Gaussian splatting [Kerbl et al., 2023], a technique characterized by remarkable results and reasonable hardware requirements. Gaussian splatting, a method for rendering radiation fields in real time, uses 3D Gaussians instead of traditional raster primitives such as triangles, enabling the creation of highly detailed and photorealistic scenes.

The pace of innovation in this field is rapid, as the timeline presented in Chapter 3 of the thesis demonstrates. New methods and techniques come onto the market almost every month, offering promising results and new possibilities. This continuous development underlines the dynamism of automatic 3D model generation and points to an exciting future in which the boundaries of digital creativity and realism will be pushed further and further.

5.2 Handle Generative Bias

Bias in generative AI has become a critical issue, including in the field of automatic 3D model generation. This bias is due in part to the 2D diffusion models that form the basis for many 3D modeling techniques or models like CLIP [Luccioni et al., 2023; Radford et al., 2021]. 2D diffusion models are commonly trained on large datasets comprised of internet-sourced images, which are often not free from societal stereotypes and biases. As a result, the distorted representations of the world inherent in these data sets are unintentionally transferred to the 3D models generated from them [Buolamwini and Gebru, 2018].

To initially explore the issue of generative bias in text-to-3D models, a preliminary experimental approach was employed. This involved using diverse human figures and roles as prompts in various text-to-3D modeling systems. It's important to note that these experiments were conducted on a limited scale due to time and computational constraints, and thus, they offer only a cursory glimpse into potential biases rather than statistically significant evidence. Early observations suggest potential biases: for instance, prompts such as "homeless person" tended to yield images of people of color more frequently, while "rich person" predominantly resulted in representations of white male figures. Similarly, gender biases were observed in occupational prompts, with professions like "teacher" often depicted as female figures and roles such as "engineer" or "CEO" primarily as male figures. These preliminary results hint at a tendency of AI systems to associate certain demographics with specific roles and socioeconomic statuses, potentially reflecting and perpetuating societal stereotypes, although these findings are not conclusive.

There are several theoretical approaches that could mitigate this problem, although their effectiveness has yet to be thoroughly tested. One such approach is the diversification of training datasets. A broader range of images that ensures a balanced representation of different demographic groups and occupations could potentially reduce bias. Another theoretical step is the implementation of algorithms that actively combat bias. These could be algorithms that are able to recognize and correct biased patterns in the generated models. Finally, a continuous evaluation and updating process for these models is proposed to adapt to changing societal norms and expectations, ensuring that the models remain relevant and unbiased over time.

While these proposed measures have not been empirically validated in the context of large-scale 3D model creation, they represent theoretical steps towards creating more

balanced and unbiased AI models. If proven effective, they could contribute significantly to a more inclusive and representative digital world [Luccioni et al., 2023].

Chapter 6

Conclusion

//TODO

6.1 Summary of Findings

This thesis has explored various methods used for automatic 3D model generation, focusing on DreamFusion, Magic3D, Fantasia3D, Magic123, and Wonder3D. Each method was not only introduced but also scrutinized through a comparative analysis. The examination revealed that DreamFusion, one of the earlier methods, tended to produce more blurred outputs compared to its successors. This could be attributed to the initial stages of development in this field. In contrast, newer models like Fantasia3D showed improvements, especially when initiated from a shape approximating the target object. Interestingly, a random sphere as a starting point was found ineffective.

A significant aspect noted was the evolution in speed and efficiency of model generation. Earlier methods required more time for results that aligned with the text prompts, whereas newer models showcased rapid generation capabilities. Despite advancements, challenges in generating intricate details, such as plants, persisted across all methods, highlighting areas for future optimization. Fantasia3D, with its advanced texture generation stage, excelled in producing realistic textures, though it struggled in overall model generation.

One of the most promising developments observed was in the newest methods, such as Genie, which demonstrated fast and detailed outputs, signaling rapid progress in the field. However, the presence of bias in generative AI was a recurring concern, underscoring the need for continued research and development to address this issue.

In a series of tasks testing the capabilities of various 3D rendering methods, distinct patterns emerged across different prompts. For the Playmobil figure, characterized by its uniform base structure, DreamFusion struggled with intricate geometry but maintained color accuracy, while Magic3D excelled in realistic textures and geometric precision for simpler, flat structures. Fantasta3D often overemphasized realism at the expense of essential prompt features, and Magic123, although capturing the essence well, added superfluous elements. Wonder3D frequently failed in even basic details. In the bread prompt, emphasizing symmetry and specific properties, DreamFusion continued to falter with structure, Magic3D showed robustness to directive prompts, and Fantasia3D was disoriented by specificity. Magic123's replication of the input image raised questions about its adaptability, and Wonder3D struggled with overall structure.

When rendering a sleeping dog, a task focusing on organic forms, texture, and anatomy, DreamFusion displayed the lowest detail level, failing to distinguish the subject. Magic3D effectively captured essential elements but lacked textural depth. Fantasia3D, focusing on realistic textures, missed basic geometric accuracy. Magic123 accurately shaped the original image but struggled with different views, while Wonder3D provided a consistent, multi-view output. Finally, in rendering a fern with complex elements and contrasting materials, DreamFusion captured basic shapes but missed finer details. Magic3D's output was disappointing for intricate objects, whereas Fantasia3D achieved high textural realism but lacked geometric accuracy. Magic123's results were basic and view-dependent, and Wonder3D, although capturing the overall form, revealed undetailed textures and floating parts upon closer examination. Across these prompts, each method displayed unique strengths and weaknesses, highlighting the challenges and potentials in automatic 3D model generation.

6.2 Contributions to the Field

This thesis has significantly contributed to the field of automatic 3D model generation. It has elucidated the basics and detailed the functionality of key methods, making the complex domain accessible to a broader audience. The work showcases realistic expectations of model generation for average users, distinct from the high-end results often illustrated in official papers, which require substantial computational resources.

A practical aspect of this thesis is the provision of a notebook facilitating the application of these methods, enabling users to assess and validate the capabilities of each method personally. Additionally, a comprehensive comparison across various methods offers insights into their strengths and weaknesses. The inclusion of Evaluate3D in the notebook provides a user-friendly tool for analyzing generated models, further enhancing the practical utility of this thesis in the field.

6.3 Implications and Practical Applications

discusses real-world impact of the findings.

Appendix A

Additional Images

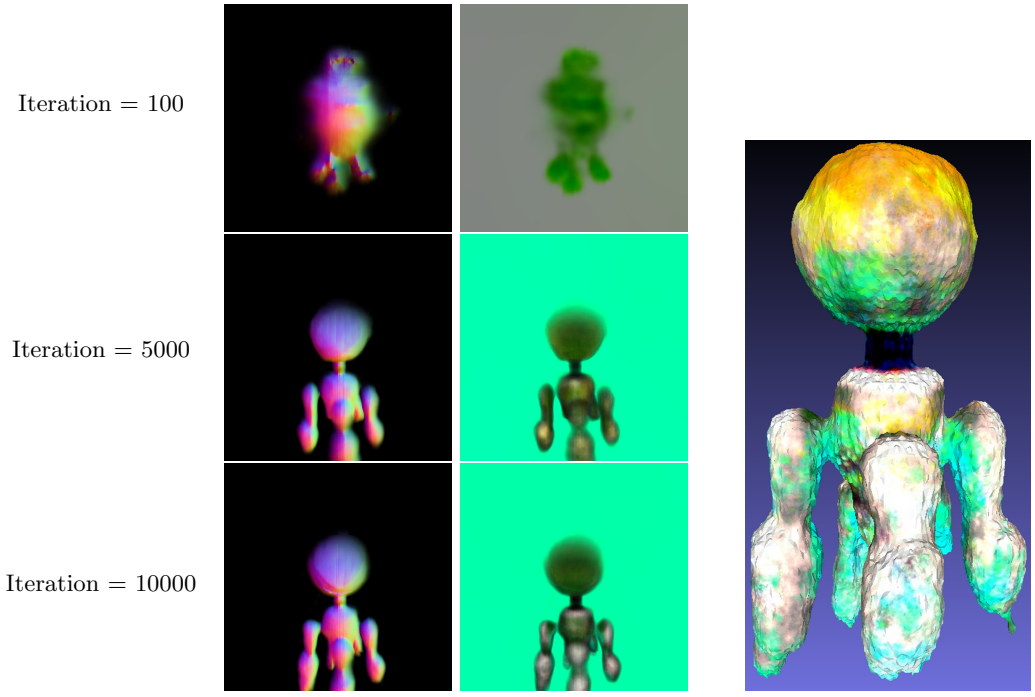


Figure 32: The generation process of DreamFusion using the prompt “a robot made out of plants” a second time.

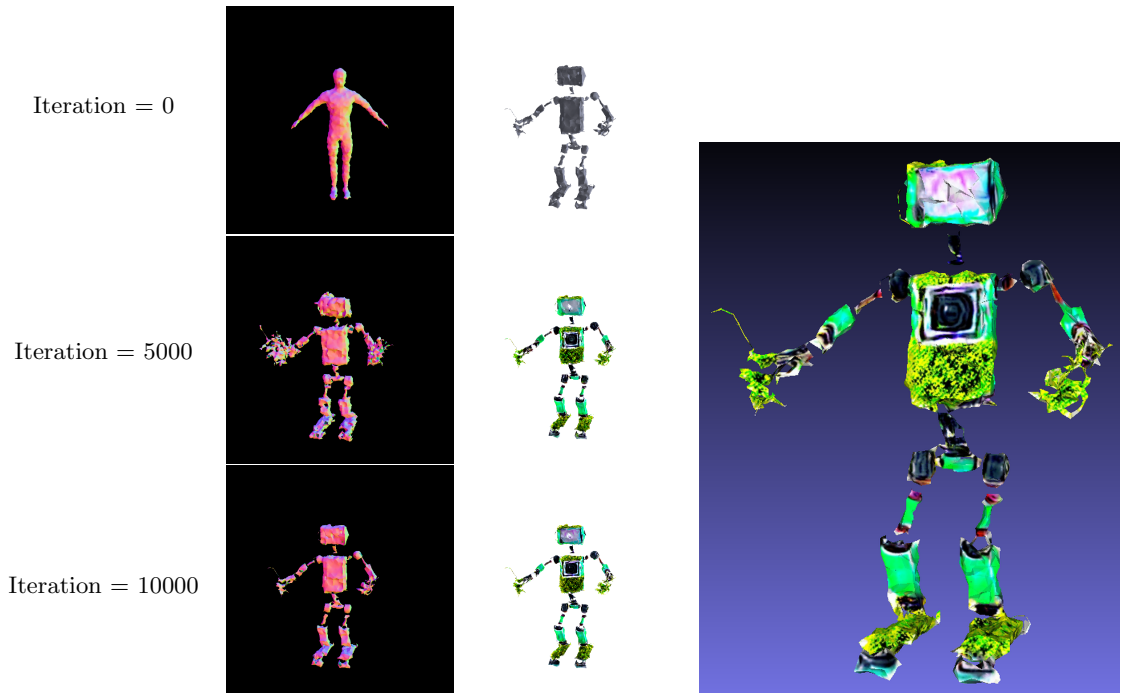


Figure 33: Initializing Fantasia3D with a coarse mesh representing a basic human figure

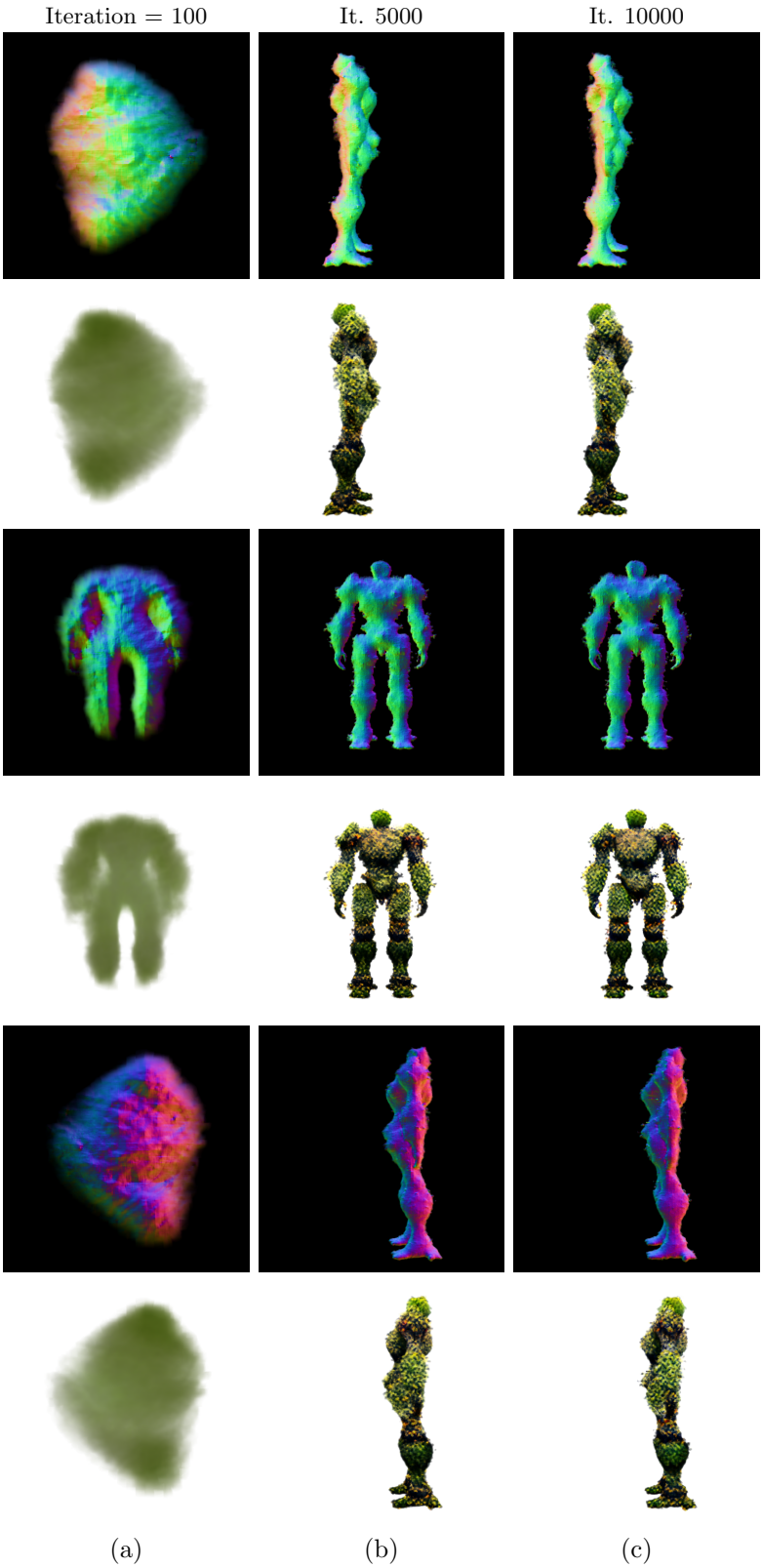


Figure 34: The coarse in Magic123; From top to bottom: right, back and left view

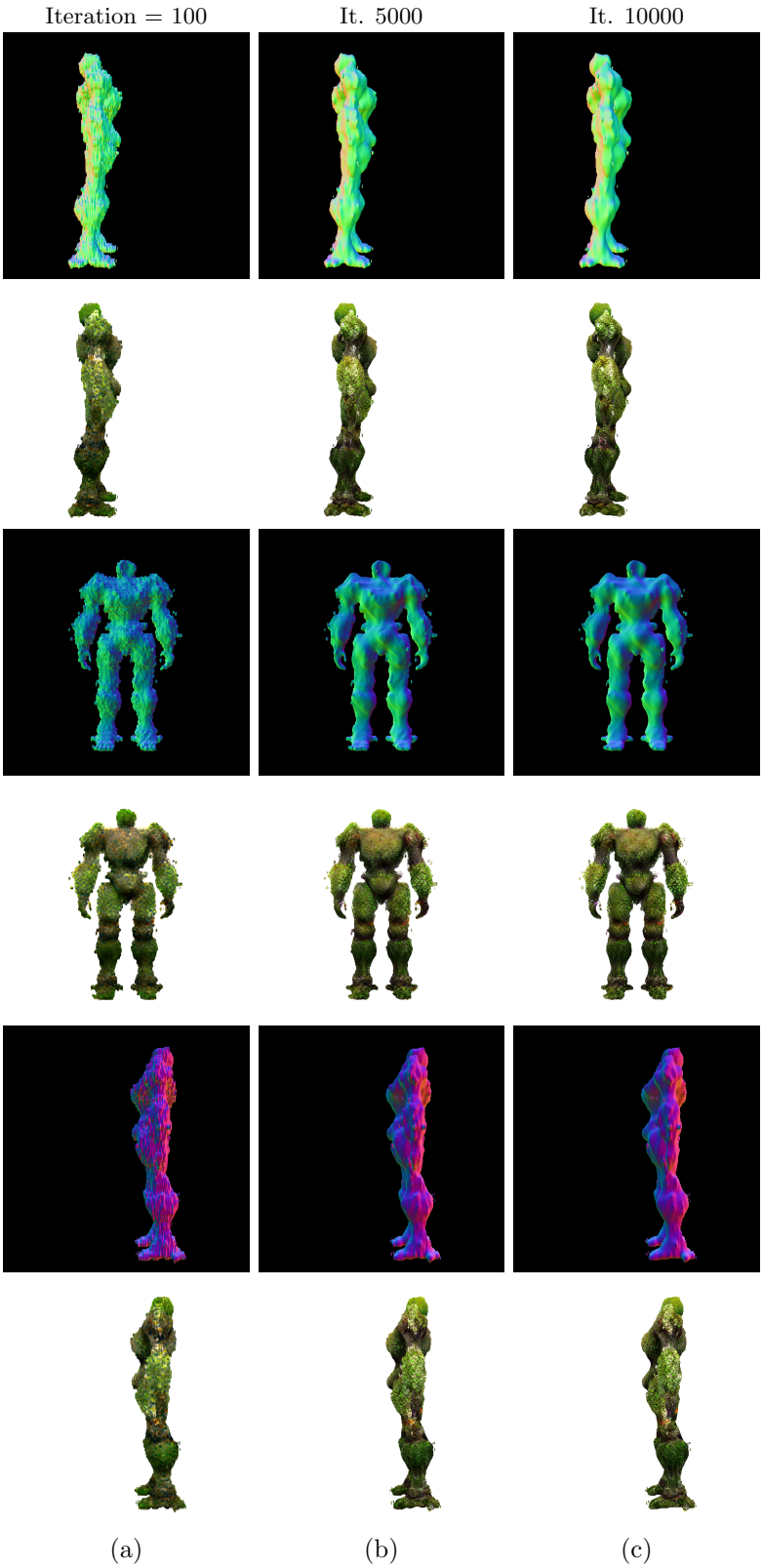
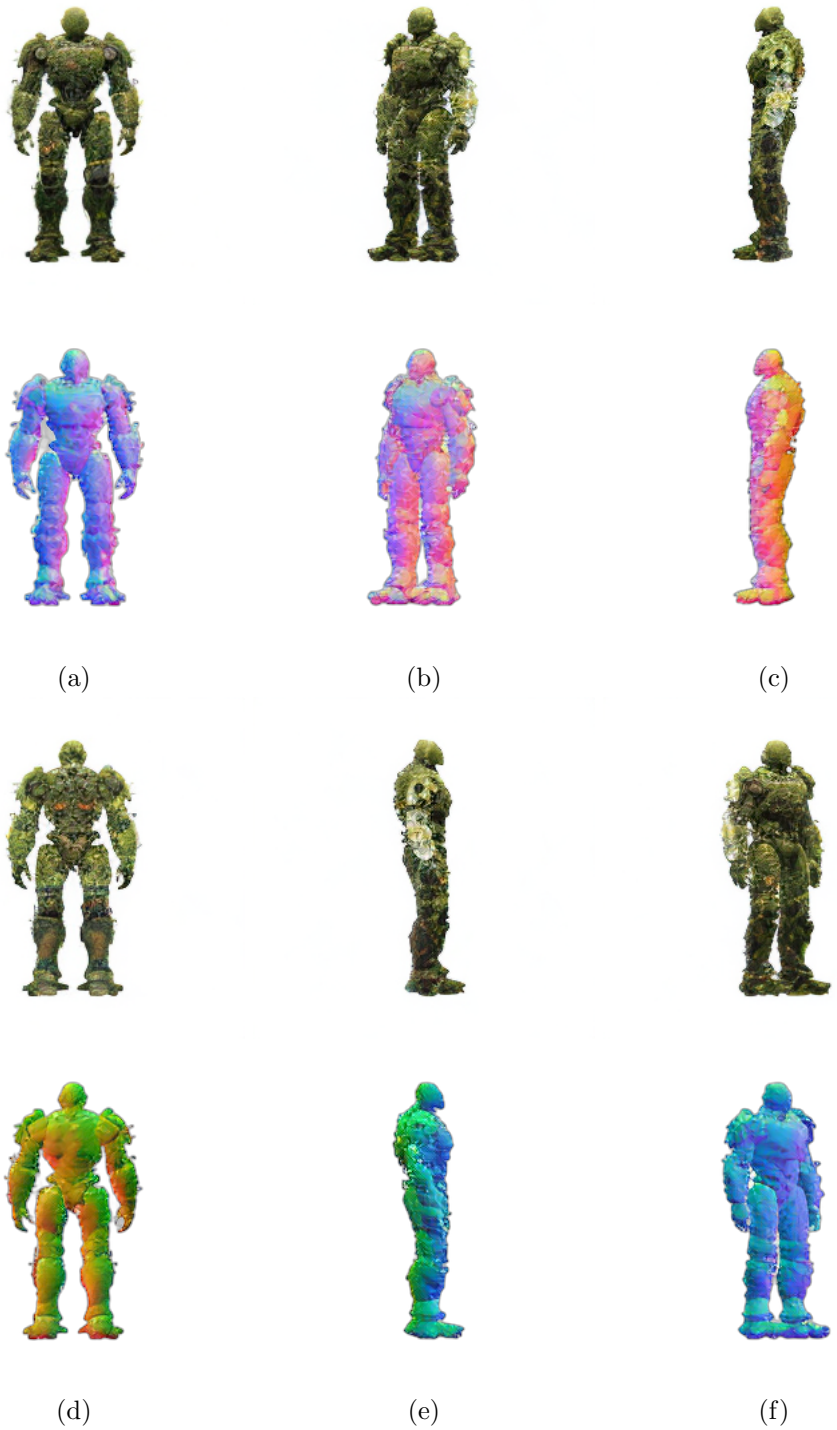


Figure 35: The refine in Magic123; From top to bottom: right, back and left view



(a) (b) (c)

(d) (e) (f)

Figure 36: Wonder3D initialization of multi-view color images and normals; (a) front, (b) front left, (c) left, (d) back, (e) right, (f) front right

Appendix B

Adaptive Modifications in Threestudio Implementations

In the Threestudio project's adaptation of Dreamfusion, Magic3D, Fantasia3D, and Magic123, several structural changes were implemented, distinguishing them from their official paper counterparts. These differences stated here are directly taken as given in the Threestudio GitHub project page [Guo et al., 2023]:

For Dreamfusion, open-source Text-to-Image models like StableDiffusion and DeepFloyd IF are used instead of Imagen as in the original paper. A lower guidance scale of 20 is applied for DeepFloyd IF, as opposed to 100 used for Imagen. The normalization of albedo color deviates from the sigmoid approach; it's simply scaled from a range of [-1,1] to [0,1] to aid in convergence. The project employs HashGrid encoding and uniform ray sampling, contrasting with Integrated Positional Encoding and MipNeRF360's sampling strategy in the paper. Camera settings and density initialization are adopted from Magic3D, which slightly differ from the DreamFusion paper. Additionally, there are variations in hyperparameters like the weighting of loss terms.

In Magic3D's implementation within Threestudio, open-source T2I models are used for the coarse stage, as opposed to eDiff-I in the paper. The guidance scale for DeepFloyd IF is set at 20, contrasting with the paper's 100 for eDiff-I. The coarse stage utilizes analytic normal instead of predicted normal and includes orientation loss as in DreamFusion, which the paper does not incorporate. There are also omissions and potential differences in aspects like the weighting of loss terms and DMTet grid resolution.

For Fantasia3D, tangent-space normal perturbation is enabled by default, diverging from the paper's approach, but can be disabled with specific system settings. Magic123 in Threestudio is an unofficial re-implementation, sharing the overall idea with the official version but differing in aspects such as hyperparameters. It doesn't support Textual Inversion, necessitating a text prompt for training. These modifications reflect Threestudio's approach to adapting and optimizing these models within their resource constraints and technological framework.

Additionally, specific technical parameters were set to streamline the generation flow of various models, occasionally diverging from the original methods to minimize computational costs and ensure operational feasibility. An example of this adaptation is seen in the Fantasia3D project [Chen et al., 2023], where the original configuration used 512×512 images for the geometry stage and scaled up to 2048×2048 for the appearance stage, a

setup requiring substantial processing power. Modifying these parameters could potentially lead to improved outcomes, but at the expense of significantly increased computational demands, a constraint particularly relevant when working with platforms like Colab. Therefore, these values were maintained as originally set. It is important to point out that the adjustments mentioned here, such as the image resolution and optimizer settings, represent only a fraction of the extensive hyperparameter changes implemented in the Threestudio versions of these models, reflecting a comprehensive optimization approach tailored to specific resource constraints.

Each model was assigned a seed value of 0 to ensure consistent reproducibility. DreamFusion operates on a 64×64 resolution, whereas Magic3D uses 64×64 in its coarse stage and 512×512 in the refine stage. Fantasia3D is set to 512×512 in both its geometry and appearance stages. Magic123 employs 128×128 images for its coarse stage and upgrades to 512×512 for the refine stage. Wonder3D, on the other hand, uses 256×256 for its coarse stage and escalates to 1024×1024 in the refine stage, indicating the highest resolution among these models. The optimizer learning rate is uniformly set at 0.01 across all models. DreamFusion, Magic3D, and Magic123 utilize the Adam optimizer, whereas Fantasia3D and Wonder3D employ the AdamW optimizer. The latter is preferred for its enhanced handling of weight decay and ability to improve model generalization. Regarding precision, all models, except Wonder3D, adopt a 16-mixed precision approach, which integrates 16-bit computational efficiency with 32-bit accuracy. Wonder3D, on the other hand, opts for 16-bit precision, prioritizing faster computation at the possible expense of precision.

Bibliography

Anderson, B. D. (1982). Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326.

Arandjelović, R. and Zisserman, A. (2021). Nerf in detail: Learning to sample for view synthesis. *arXiv preprint arXiv:2106.05264*.

Balaji, Y., Nah, S., Huang, X., Vahdat, A., Song, J., Zhang, Q., Kreis, K., Aittala, M., Aila, T., Laine, S., Catanzaro, B., Karras, T., and Liu, M.-Y. (2022). ediff-i: Text-to-image diffusion models with ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*.

Barron, J. T., Mildenhall, B., Verbin, D., Srinivasan, P. P., and Hedman, P. (2022). Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479.

Betker, J., Goh, G., Jing, L., Ramesh, A., et al. (2023). Improving image generation with better captions. `https://cdn.openai.com/papers/dall-e-3.pdf`. Accessed: [25.11.2023].

Brophy, E., Wang, Z., She, Q., and Ward, T. (2023). Generative adversarial networks in time series: A systematic literature review. *ACM Computing Surveys*, 55(10):1–31.

Buolamwini, J. and Gebru, T. (2018). Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*, pages 77–91. PMLR.

Chen, R., Chen, Y., Jiao, N., and Jia, K. (2023). Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. GitHub: `https://github.com/Gorilla-Lab-SCUT/Fantasia3D`.

Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., and Ranzuglia, G. (2008). MeshLab: an Open-Source Mesh Processing Tool. In Scarano, V., Chiara, R. D., and Erra, U., editors, *Eurographics Italian Chapter Conference*. The Eurographics Association.

Discord Inc. (2023). Discord. `https://discord.com`. Accessed: [2023-12-06].

Doersch, C. (2016). Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, 63(11):139–144.

Goodfellow, I. J., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press, Cambridge, USA. `http://www.deeplearningbook.org`.

Google LLC (2023). Google colaboratory. `https://colab.research.google.com/`. Accessed: [1.11.2023].

Guo, Y.-C., Liu, Y.-T., Shao, R., Laforte, C., Voleti, V., Luo, G., Chen, C.-H., Zou, Z.-X., Wang, C., Cao, Y.-P., and Zhang, S.-H. (2023). threestudio: A unified framework for 3d content generation. `https://github.com/threestudio-project/threestudio`.

Haggerty, D. et al. (2019). trimesh. `https://trimsh.org/`. Accessed: [2019-12-8].

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. (2017). beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851.

Ho, J. and Salimans, T. (2022). Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*.

Hu, S., Zhou, K., Li, K., Yu, L., Hong, L., Hu, T., Li, Z., Lee, G. H., and Liu, Z. (2023). Consistentnerf: Enhancing neural radiance fields with 3d consistency for sparse view synthesis. *arXiv preprint arXiv:2305.11031*.

Hyvärinen, A. and Dayan, P. (2005). Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4).

Jain, Yu, and et al. (2023). Genie. `https://lumalabs.ai/genie`. Accessed: [2023-12-06].

Kerbl, B., Kopanas, G., Leimkühler, T., and Drettakis, G. (2023). 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4).

Kingma, D., Salimans, T., Poole, B., and Ho, J. (2021). Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Lahav, A. and Tal, A. (2020). Meshwalker: Deep mesh understanding by random walks. *ACM Transactions on Graphics (TOG)*, 39(6):1–13.

Lin, C.-H., Gao, J., Tang, L., Takikawa, T., Zeng, X., Huang, X., Kreis, K., Fidler, S., Liu, M.-Y., and Lin, T.-Y. (2023). Magic3d: High-resolution text-to-3d content creation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 300–309.

Liu, Q., Lee, J., and Jordan, M. (2016). A kernelized stein discrepancy for goodness-of-fit tests. In Balcan, M. F. and Weinberger, K. Q., editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 276–284, New York, USA. PMLR.

Liu, R., Wu, R., Van Hoorick, B., Tokmakov, P., Zakharov, S., and Vondrick, C. (2023). Zero-1-to-3: Zero-shot one image to 3d object. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9298–9309.

Long, X., Guo, Y.-C., Lin, C., Liu, Y., Dou, Z., Liu, L., Ma, Y., Zhang, S.-H., Habermann, M., Theobalt, C., et al. (2023). Wonder3d: Single image to 3d using cross-domain diffusion. *arXiv preprint arXiv:2310.15008*.

Luccioni, A. S., Akiki, C., Mitchell, M., and Jernite, Y. (2023). Stable bias: Analyzing societal representations in diffusion models. *arXiv preprint arXiv:2303.11408*.

Martínez, G., Watson, L., Reviriego, P., Hernández, J. A., Juarez, M., and Sarkar, R. (2023). Towards understanding the interplay of generative artificial intelligence and the internet. *arXiv preprint arXiv:2306.06130*.

McAuley, S., Hill, S., Hoffman, N., Gotanda, Y., Smits, B., Burley, B., and Martinez, A. (2012). Practical physically-based shading in film and game production. In *ACM SIGGRAPH 2012 Courses*, SIGGRAPH '12, New York, USA. Association for Computing Machinery.

Michalkiewicz, M., Pontes, J. K., Jack, D., Baktashmotlagh, M., and Eriksson, A. (2019). Deep level sets: Implicit surface representations for 3d shape inference. *arXiv preprint arXiv:1901.06802*.

Michelucci, U. (2022). An introduction to autoencoders. *arXiv preprint arXiv:2201.03898*.

Midjourney, I. (2023). Midjourney. `https://docs.midjourney.com`. Accessed: [2023-12-06].

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2021). Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106.

Mordvintsev, A., Pezzotti, N., Schubert, L., and Olah, C. (2018). Differentiable image parameterizations. *Distill*. https://distill.pub/2018/differentiable-parameterizations.

Müller, T., Evans, A., Schied, C., and Keller, A. (2022). Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15.

Murtagh, F. (1991). Multilayer perceptrons for classification and regression. *Neurocomputing*, 2(5):183–197.

Noriega, L. (2005). Multilayer perceptron tutorial. *School of Computing. Staffordshire University*, 4(5):444.

Poole, B., Jain, A., Barron, J. T., and Mildenhall, B. (2022). Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*.

Qian, G., Mai, J., Hamdi, A., Ren, J., Siarohin, A., Li, B., Lee, H.-Y., Skorokhodov, I., Wonka, P., Tulyakov, S., et al. (2023). Magic123: One image to high-quality 3d object generation using both 2d and 3d diffusion priors. *arXiv preprint arXiv:2306.17843*.

Rabby, A. and Zhang, C. (2023). Beyondpixels: A comprehensive review of the evolution of neural radiance fields. *arXiv preprint arXiv:2306.03000*.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., et al. (2023). Dall-e 3. `https://openai.com/dall-e-3`. Accessed: [25.11.2023].

Ranftl, R., Bochkovskiy, A., and Koltun, V. (2021). Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12179–12188.

Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., and Koltun, V. (2020). Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE transactions on pattern analysis and machine intelligence*, 44(3):1623–1637.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In Xing, E. P. and Jebara, T., editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1278–1286, Bejing, China. PMLR.

Roberts, G. O. and Tweedie, R. L. (1996). Exponential convergence of langevin distributions and their discrete approximations. *Bernoulli*, pages 341–363.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695.

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E. L., Ghasemipour, K., Gontijo Lopes, R., Karagol Ayan, B., Salimans, T., et al. (2022). Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X., and Chen, X. (2016). Improved techniques for training gans. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Shen, T., Gao, J., Yin, K., Liu, M.-Y., and Fidler, S. (2021). Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. *Advances in Neural Information Processing Systems*, 34:6087–6101.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR.

Song, Y. (2021). Score-based generative modeling through stochastic differential equations. `https://yang-song.net/blog/2021/score/`. Accessed: [26.11.2023].

Song, Y., Durkan, C., Murray, I., and Ermon, S. (2021). Maximum likelihood training of score-based diffusion models. *Advances in Neural Information Processing Systems*, 34:1415–1428.

Song, Y. and Ermon, S. (2019). Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2020). Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*.

Tang, J. (2022). Stable-dreamfusion: Text-to-3d with stable-diffusion. https://github.com/ashawkey/stable-dreamfusion.

Xiao, Z., Kreis, K., and Vahdat, A. (2021). Tackling the generative learning trilemma with denoising diffusion gans. *arXiv preprint arXiv:2112.07804*.

Xu, Y., Tong, X., and Stilla, U. (2021). Voxel-based representation of 3d point clouds: Methods, applications, and its potential use in the construction industry. *Automation in Construction*, 126:103675.

Yang, L., Zhang, Z., Song, Y., Hong, S., Xu, R., Zhao, Y., Shao, Y., Zhang, W., Cui, B., and Yang, M.-H. (2022). Diffusion models: A comprehensive survey of methods and applications. *arXiv preprint arXiv:2209.00796*.

Zhang, H., Wang, C., Tian, S., Lu, B., Zhang, L., Ning, X., and Bai, X. (2023). Deep learning-based 3d point cloud classification: A systematic survey and outlook. *Displays*, 79:102456.

Erklärung

Ich erkläre hiermit gemäß §9 Abs. 12 APO, dass ich die vorstehende Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Des Weiteren erkläre ich, dass Inhalt und Wortlaut der beiden Fassungen (digital/in Papierform) identisch sind und zur Kenntnis genommen wurde, dass die digitale Fassung einer durch Software unterstützten, anonymisierten Prüfung auf Plagiate unterzogen werden kann

Datum	Unterschrift