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 continued emergence of generative models, deep-learning architectures, and data-driven methods has opened a new era of technological opportunity, particularly in the area of automated 3D model generation. Such advances have become increasingly important in a number of sectors, including gaming, virtual reality, medicine and architecture. Given the increasing demand for 3D objects in these industries, a comprehensive analysis of the underlying technologies is of paramount importance. This thesis attempts to fill this scientific gap by providing a systematic examination of the various methods for automatic 3D model generation.

Based on generative algorithms, machine learning and artificial intelligence, the study examines various techniques and methods that have gained acceptance in this field. The goal is to objectively evaluate their efficiency in creating 3D models that are not only accurate but also visually compelling. This analytical framework is focused on evaluating methods such as Variational Autoencoders (VAEs), Generative Adversarial Networks (GANs), Diffusion Models, and Neural Radiance Fields (NeRFs), among others.

The study employs a carefully designed experimental setup and uses specified performance metrics to quantitatively and qualitatively analyze the strengths and weaknesses of these methods. Through this critical evaluation, the thesis aims to provide both an academic resource and a practical guide for subsequent research and applications in the field of automated 3D model generation. Thus, the contributions of this thesis go beyond a purely academic investigation; it serves as a foundational resource that can both deepen existing understanding and inform future computer graphics methodology.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In today's digital landscape, the demand for 3D models is steadily increasing, driven by the need for immersive and realistic visual experiences. Researchers and practitioners have leveraged generative AI techniques to develop innovative methods that automate the process of creating 3D models. These innovations have the potential to reshape how we interact with digital environments and facilitate the simulation, analysis, and visualization 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 **??** may appear straightforward, acquiring a more in-depth understanding of the diverse methodologies and their foundational principles greatly enriches the learning process. This deeper insight not only improves the practical application of these models but also opens up possibilities for further advancements in the field of automatic 3D model generation.

This thesis provides a comprehensive examination of advances in automated 3D model generation. It delves into a variety of models and technologies, 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 significantly 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. The insights gained from this comparative study aim to inspire further innovation in this rapidly evolving field, pushing the boundaries of what is possible in 3D modeling and opening up new avenues of exploration and discovery in 3D synthesis.

To provide a foundation for this exploration, a comprehensive examination of the fundamentals of 2D generative AI is undertaken. Essential generative models such as Variational Autoencoders (VAEs) [**??**], Generative Adversarial Networks (GANs) [**?**] and Diffusion Models [**???**] are explained. Additionally, a brief introduction in Contrastive Language-Image Pre-training (CLIP) [**?**] and various forms of 3D data representation is given.

The research further evaluates different approaches to generating 3D models, including methods based on images and text input. Each method presents unique challenges and opportunities, and their results are thoroughly examined through a comparative study. This investigation incorporates well-defined experimental setups and the application of rigorous performance metrics, such as accuracy, efficiency, and realism, to conduct comprehensive

results analyses.

Additionally, this thesis addresses future opportunities in the field of 3D modeling. It highlights emerging trends and potential research directions that promise to redefine the 3D modeling landscape. The practical implications of these findings are also considered, underscoring their significance in real-world applications.

By providing a nuanced and detailed exploration of automated 3D model generation, this thesis contributes to the understanding and advancement of this dynamic and impactful field.

//TODO briefly explain test metrics / how evaluated.

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.

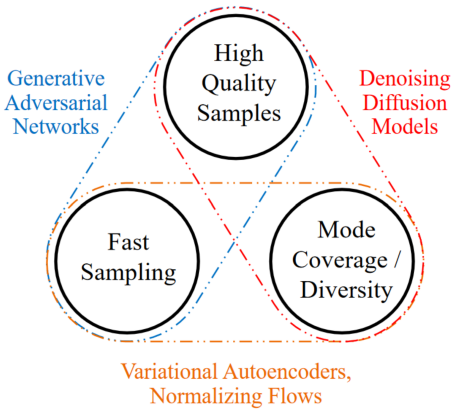


Figure 1: The Generative Learning Trilemma: Balancing Quality, Speed, and Diversity in Generative Models. [**?**]

The chapter starts with a brief overview of the most common generative text-to-2D models, as they play a central role in the process of 3D synthesis. The section aims to shed light on the "generative learning trilemma", a concept introduced in Figure **??** by **?** that describes the balance between quality, speed and diversity in different generative models. The chapter includes Variational Autoencoders (VAEs) [**??**], 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) [**?**] are presented, highlighting their fast training mechanisms for producing high-quality samples that include both generator and discriminator components. 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) [**??**], while Stochastic Gradient Methods (SGMs) [**?**] and Stochastic Differential Equations (SDEs) [**??**] are also addressed.

Furthermore, the chapter deals with Contrastive Language-Image Pre-training (CLIP) [?] and illustrates its effectiveness in bridging the gap between natural language and visual data, which is crucial for text-driven 3D model generation. 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 models, 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 facilitates 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" [??], 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 ?, ?, and ?, strikes a balance between data compression and information preservation.

However, a notable limitation arises when the encoder and decoder become too closely matched. In such instances, the latent space becomes less controllable, leading to challenges in interpretability and regularity [?]. 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 [?]. 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 [?]. 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. [**?**].

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

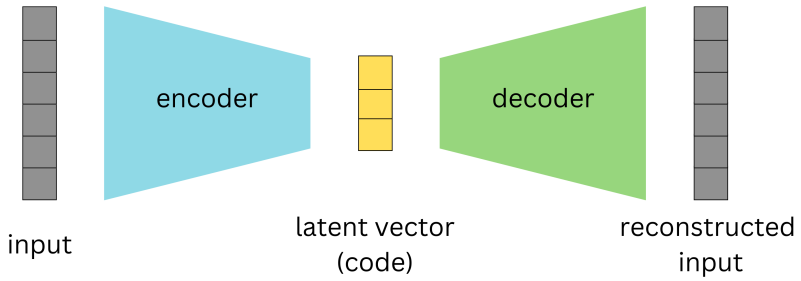


Figure 2: Schematic of an Autoencoder: Demonstrating the process of dimensionality reduction to a latent vector and subsequent reconstruction, aiming to minimize reconstruction loss. TODO ADD LOSS https://towardsdatascience.com/difference-between-autoencoder-ae-and-variational-autoencoder-vae-ed7be1c038f2

At the core of Variational Autoencoders is the handling of the latent space. Unlike traditional autoencoders that encode data to a single point, VAEs employ a probabilistic approach, encoding data into a distribution characterized by parameters μ (mean) and σ (standard deviation). This approach not only allows for manipulation of data within the latent space but also paves the way for data generation. VAEs address 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, instead of encoding to a single deterministic point [**?**]. 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.

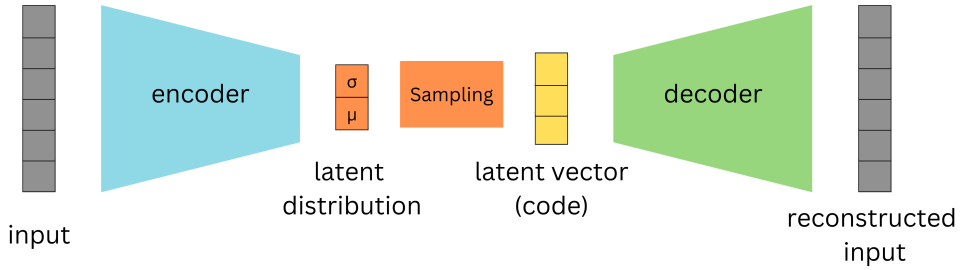


Figure 3: Functionality of a Variational Autoencoder: This figure illustrates the incorporation of a latent distribution, characterized by mean and standard deviation, for enhancing latent space regularization, enabling more effective and diverse data generation.

Despite their advanced capabilities, VAEs have some limitations. A common observation, as noted by **?**, 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 and other less distinct points, resulting in blurry images [**?**]. 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 [**?**]. 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 [**??**].

2.2 Generative Adversarial Networks – GANs

"When a deep neural network is used to generate data, the corresponding density function may be computationally intractable" [**?**]. 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 [**?**]. 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" [**?**]. 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.

In the initial training phase, the discriminator is trained on a dataset of real, unlabeled

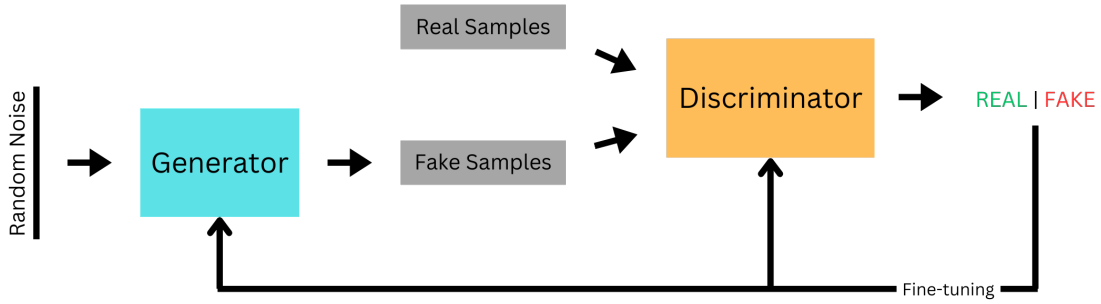


Figure 4: Schematic representation of Generative Adversarial Networks showcasing the interaction between the generator and discriminator networks in generating new data.

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 [**?**].

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 [**?**]. 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 [**?**],

Nevertheless, GANs pose a substantial challenge in their training process as they are hard to train [**?**]. In addition, **?** 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 [**?**]. 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 samples [**?**]. 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 [**?**]. 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 [**?**].

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). Specifically, they address the limitations in data quality and generation efficiency that VAEs and GANs encounter. 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.

? distingueshe between three main approaches that dominate the study of diffusion models, which are going to be discussed shortly: Denoising Diffusion Probabilistic Models (DDPMs) [**??**], Score-based Generative Models (SGMs) [**?**], and Stochastic Differential Equations (Score SDEs) [**??**]. 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 [**?**]. These processes are illustrated in Figures **??** and **??**.

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 $\beta_1, \ldots, \beta_T$" **?**. 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 [**??**].

The mathematical formulation of the forward process is given by **?**:

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

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 [**??**]. 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 [**?**]. 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 **?**:

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

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 [**?**]. 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.

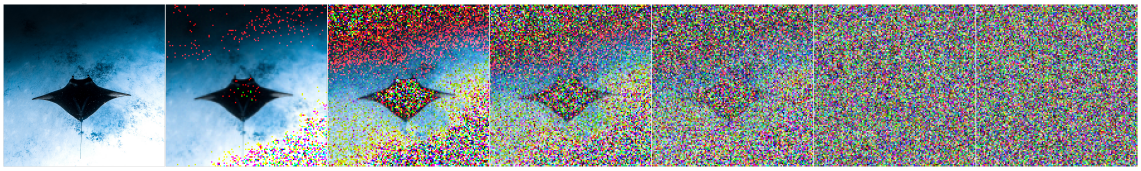


Figure 5: Illustration of the Forward Diffusion Process in DDPMs: This figure demonstrates the gradual addition of Gaussian noise to an image over multiple steps. Each subsequent image from left to right shows an increased level of noise, culminating in the far-right image, which represents a state of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

The reverse diffusion process, illustrated in Figure **??**, 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 **?**. 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 [**?**].

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

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

The latter function $p_\theta(x_{0:T})$ is also taken from **?** 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 [**??**]. 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 [**?**]. 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 [**?**].

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

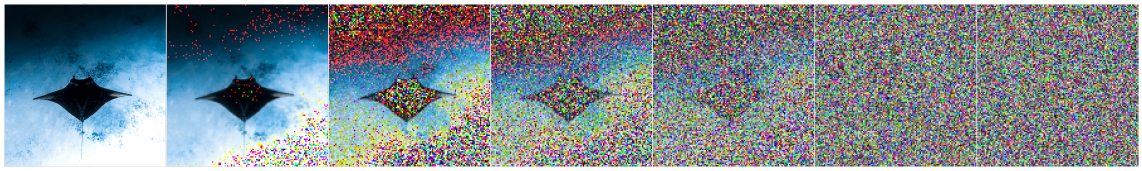


Figure 6: Visual Representation of the Reverse Diffusion Process in DDPMs: This figure illustrates the progressive removal of noise from a noisy state (right) back to the original or newly generated image (left), demonstrating the model's capability to reconstruct or create images by reversing the noise addition process.

"a Markov process [that] has to be simulated at each generation step, which greatly slows down the process" [**?**].

2.3.2 Score-Based Generative Models

In the domain of generative modeling, Score-Based Generative Models (SGMs), as introduced by **?** distinguish themselves by prioritizing the learning of a score function using the Stein score [**?**]. The score is "the gradient of the log-density function at the input data point" [**?**], 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 [**?**]. Training of such models is achieved by "minimizing the Fischer divergence between the model and the data distributions" **?**, 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 [**?**]. 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 [**?**].

In the generation of new samples, SGMs start with a random or noise image. Using Langevin dynamics, an iterative process described by [**?**], the model samples from a distribution $p(x)$ using its score function. Following **?**, 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 [**??**].

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 [**?**]. To mitigate this, **?** 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 [**?**].

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 [**?**]. 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 [**?**].

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

? 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 **?**:

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

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) [**?**]. 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 [**?**] 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:" [**?**].

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

This equation by **?** 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" [**?**]. 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 [**?**], which involves fine-tuning the model to closely approximate these score functions across a spectrum of noise levels [**?**].

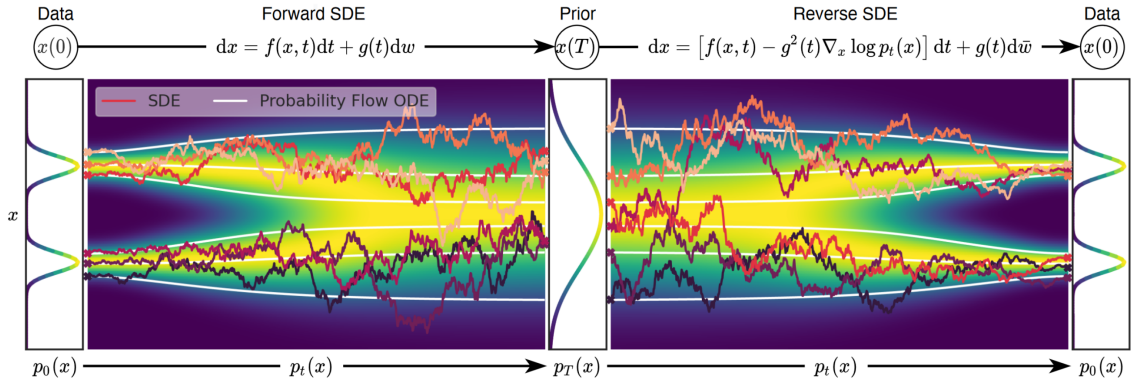


Figure 7: This figure illustrates the two-fold process in score-based generative modeling through SDEs. On the left, the Forward SDE represents the gradual transformation of data into noise, guided by the drift and diffusion coefficients. On the right, the Reverse SDE depicts the process of reconstituting 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

? 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" [**?**]. 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 a large corpus 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. CLIP is built on a transformer-based architecture, which has proven highly effective for natural language processing tasks [**?**]. It consists of two main components: an image encoder and a language encoder. The image encoder processes images using a modified version of ResNet50 [**?**] or as a second approach was build upon the Vision Transformer (ViT) [**?**], while the language encoder uses another modified transformer-based model to process textual descriptions [**?**]. By learning to associate images and text, CLIP acquires a generalized understanding of visual concepts and language semantics.

One of the remarkable aspects of CLIP is the ability for zero-shot capability. It can perform tasks without task-specific training [**?**]. 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" [**?**]. 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 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 [**??**]. 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 [**??**]. 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 [**?**]. 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 [**?**]. 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 [**?**]

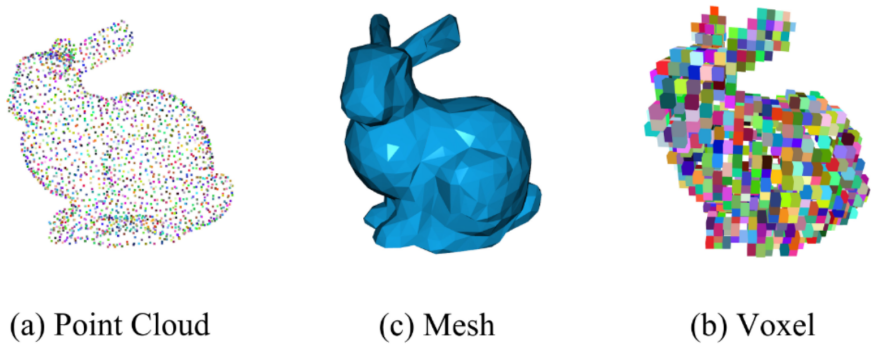
2.6 Representation forms of 3D Data

2.6.1 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 [**??**].

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 [**??**].

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 [**??**].



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

Figure 8: Representation forms of 3D data, adapted from **?**.

2.6.2 Neural Radiance Fields – NeRFs

Neural Radiance Fields (NeRFs), as introduced by **?**, 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 [**?**].

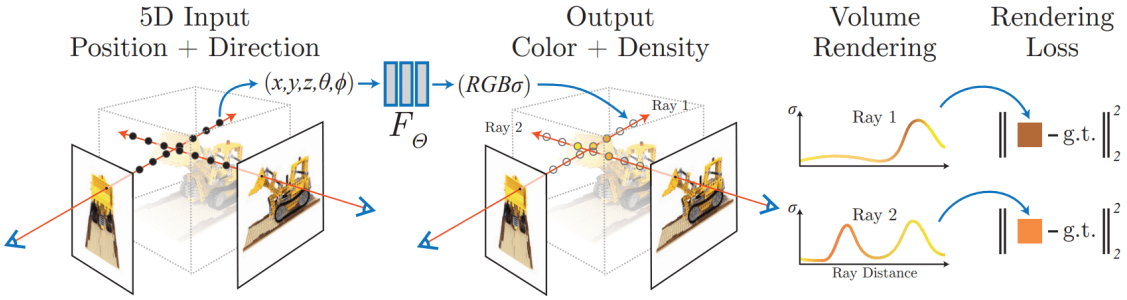


Figure 9: Sumamrized workflow of a NeRF: 5D input (Position + Direction) is processed by the MLP F_θ to output color and density. Volume rendering integrates predictions along rays to generate an image from the specified viewpoint. The rendering loss, comparing the rendered and actual images, guides the network's training and refinement process **?**

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 [**?**]. 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 [**?**].

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 **?**, 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 **??**, these density and color curves can be visualized to illustrate how density and color vary along the ray's path [**?**]. 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 [**?**]. 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 iteratively 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 [**?**].

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 [**?**]. 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 [**?**].

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 [**?**]. 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 [**?**]. 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 [**?**].

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 [**?**]. In addition, NeRF rendering is prone to sampling and aliasing issues that can lead to significant artifacts in the synthesized images [**?**]. 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 [**?**].

2.6.3 Deep Marching tetraheda – DMTet

In 3D model generation, there are two primary methods: implicit and explicit 3D representations [**?**]. Implicit methods, like Neuronal Radiance Fields (NeRFs) [**?**], 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, voxels, or point clouds. 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 [**?**].

DMTet, or Deep Marching Tetrahedra [**?**], 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 [...]" [**?**]. DMTet uniquely combines the benefits of both implicit and explicit 3D representations, leveraging a novel hybrid 3D representation approach.

The method propesed by **?** 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 [**?**]. 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" [?]. The final step involves refining the mesh into "a parameterized surface with a differentiable surface subdivision module" [?].

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

2.6.4 Instant Neual Graphics Primitives

? 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) [?]. 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" [?]. This process includes hashing, feature vector lookup, linear interpolation, and concatenation with viewing direction parameters, crucial for rendering RGB and density predictions effectively [?]. 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 [?].

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 [?].

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

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.

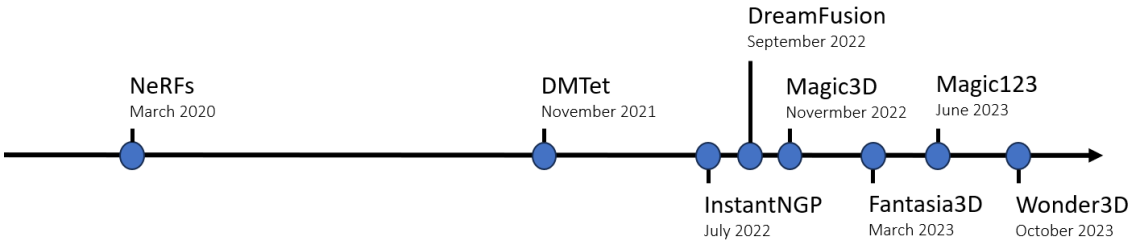


Figure 10: Timeline of generative 3D modeling technologies: This figure outlines important milestones in the development of the key methods discussed in this thesis.

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 **?**, 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 [**?**]. 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 [**?**]. 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 **??**. 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 [**?**], an imporved version of Neural Radiance Field, the mip-NeRF 360 [**?**] "that reduces aliasing" [**?**], and the Score Distillation Sampling (SDS) for the loss function.

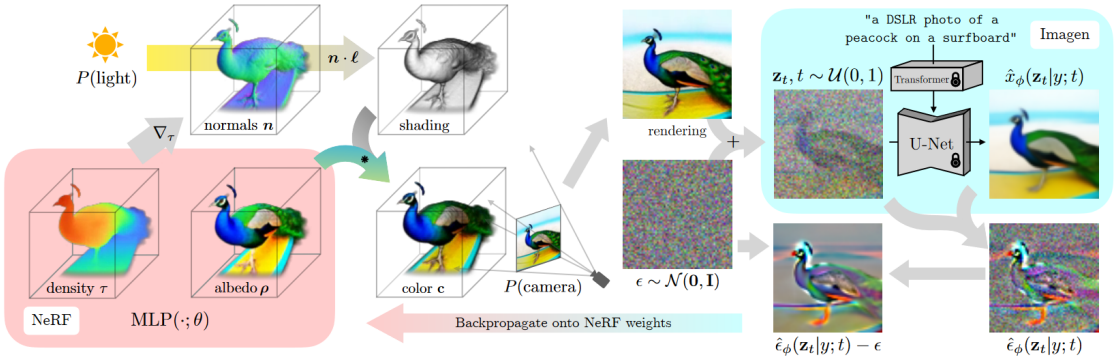


Figure 11: Overview of DreamFusion's process for transforming text into 3D models, illustrating the integration of Neural Radiance Fields, Score Distillation Sampling, and diffusion models. Image adapted from [**?**].

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 [**?**]. 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.

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 [**?**]. 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 [**?**]. 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 **?**, 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 [**?**]. Furthermore, while SDS is an effective loss function, it sometimes leads to "oversaturated and oversmoothed results [...]" [**?**]. 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"[**?**]. 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" [**?**]. 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 **?**, 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 [?]. This approach allows Magic3D to generate detailed 3D models with enhanced texture quality in a comparatively shorter time frame.

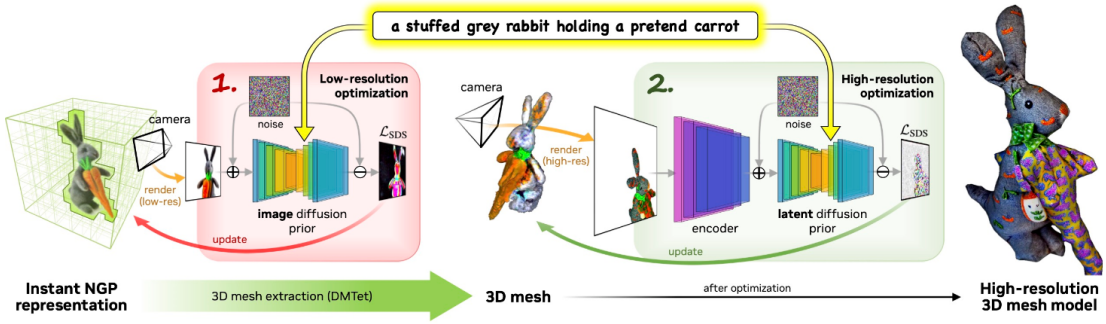


Figure 12: This illustration from [?] showcases the Magic3D process, beginning with the InstantNGP for initial 3D representation. It then details the coarse-to-fine procedure, evolving to a refined high-resolution 3D mesh model.

Magic3D's process of generating 3D models from text prompts begins with an initial stage that employs the eDiff-I base diffusion model [?], akin to Google's Imagen [?]. 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 [?]. During this phase, the model uses a sparse 3D hash grid structure from Instant NGP [?], "which allows us to represent high-frequency details at a much lower computational cost" [?]. 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 [?]. This approach not only speeds up the optimization process but also ensures the foundational quality of the coarse 3D model [?]. 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 evaluates and compares the generated image against a target image, enabling 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 [??]. 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 [?],

operates at a high resolution of 512×512, allowing for more detail of the final model [?]. 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 [?].

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" [?], 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 [?].

3.1.3 Fantasia3D

The model proposed by ? 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" [?]

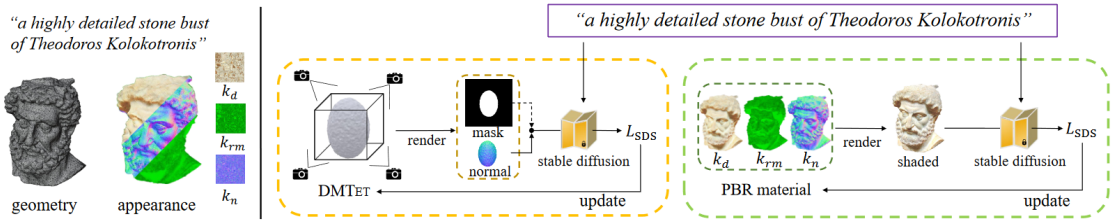


Figure 13: Overview of Fantasia3D's workflow, disentangling geometry from appearance modeling and iteratively enhancing the quality using a refinment process [?].

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 [?]. The default initialization of DMTet is an ellipsoid, but the model also accepts custom inputs.

Appearance Modeling involves training another MLP Γ, which is responsible for applying the Bidirectional Reflectance Distribution Function (BRDF) [?] to a pre-learned DMTet. This function is crucial for "predict[ing] parameters of surface material and supports high-quality 3D generation via photorealistic rendering" [?]. The BRDF focuses on

the diffuse parameter k_d, the combined roughness and metallic parameter k_{rm}, and the normal variation parameter k_n to achieve accurate shading of the geometry. These are predicted using the formula $(k_d, k_{rm}, k_n) = \Gamma(\beta(p); \gamma)$ by **?**, where $\beta(p)$ represents the surface properties at point p and γ denotes network parameters.

Both MLPs (Γ for appearance and Ψ for geometry) undergo a refinement process, using a pre-trained Stable Diffusion model [**?**]. 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 [**?**] for optimization. In this process, SDS loss functions by comparing the true image with the one generated by Fantasia3D. This comparison is achieved through ray casting, where rays are projected for each pixel in the scene. The model then renders the color of each ray, effectively translating the 3D model into a 2D image. 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 [**?**]. 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 [**?**]. Furthermore, the model is currently not able to generate complete scenes as focus is currently lying on individual object generation [**?**].

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" [**?**] - 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 [**?**].

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

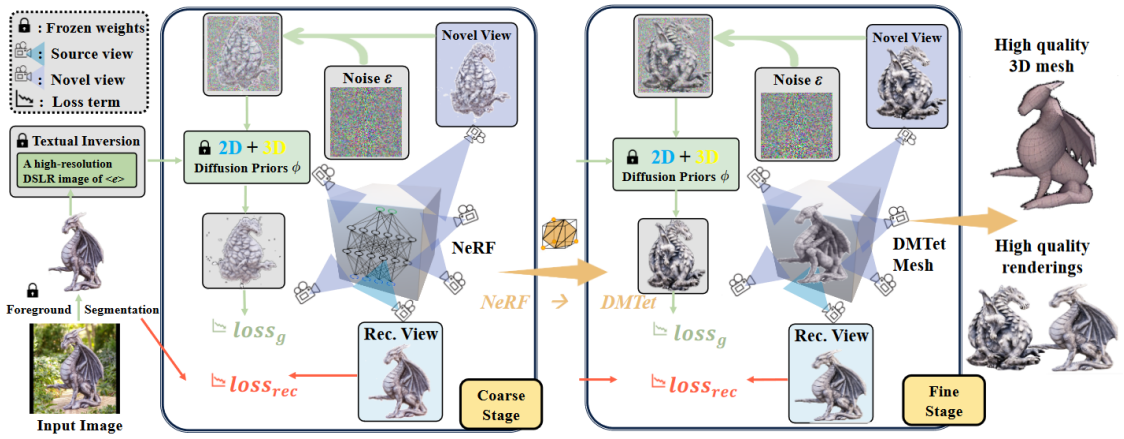


Figure 14: Overview of Magic123's two-stage process, which starts with the initial image and illustrates the transition from coarse geometry capture with Instant-NGP to high-resolution mesh refinement with DMTet [?].

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 [?]. 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 [?]. 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 [?].

The fine stage aims to refine the 3D model into a high-resolution mesh using Deep Marching Tetrahedra (DMTet) [??]. 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 [?], where both significantly enhance the level of detail and realism in the final model [?].

Both stages of Magic123 leverage a combination of 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 [?]. 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. [?].

Incorporation a trade-off parameter between the 2D and 3D priors is what defines Magic123. This parameter allows for a controlled balance between the exploration offered by 2D priors and the exploitation inherent in 3D priors, culminating in a more balanced and realistic 3D reconstruction [?].

3.2.2 Wonder 3D

? 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" [?].

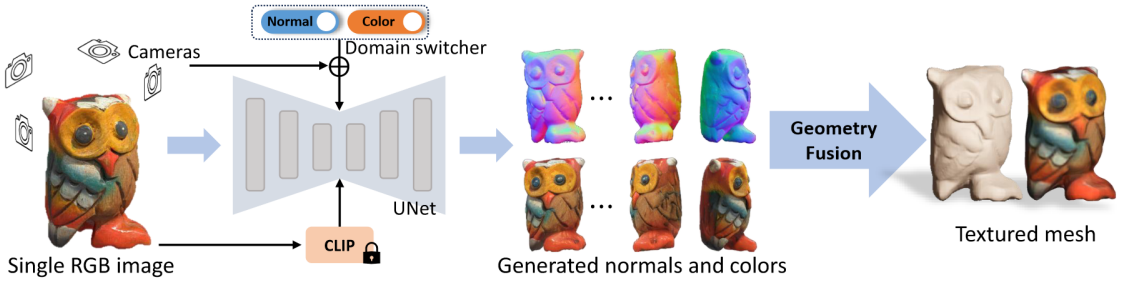


Figure 15: Summarized functionality of Wonder3D, illustrating its unique approach in generating high-fidelity textured meshes from single images using cross-domain diffusion models [?].

The process of generating 3D models from images begins with the use of CLIP [?] 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" [?].

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 ? 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. ? 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 [?]. 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 [?].

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 [**?**].

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

This equation by **?** 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 [**?**].

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 [...]" [**?**]. 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.

? 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 [**?**]. L_{rgb} ensures color accuracy, and L_{mask} calculates the errors between masks [**?**]. 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 [**?**].

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 [**?**]. While increasing the number of views could potentially enhance the detail and quality of the output, it would also significantly escalate computational demands [**?**].

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 project Threestudio [**?**] 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 [**?**], for example, was originally tested on a V100 GPU with about 32 GB RAM, but Threestudio's version also works on a T4 GPU with about 15 GB 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 specific modifications can be found in their GitHub documentation [**?**].

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 8 GB of RAM. Therefore, Google Colab [**?**] was used to mitigate these restrictions as it provides access to free GPUs 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 [**?**] 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 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 [**?**]. 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

initialization, 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 **??** and Figure **??**, both of which are included in the Appendix. For the purposes of this research, the stable-DreamFusion variant [**?**], 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.

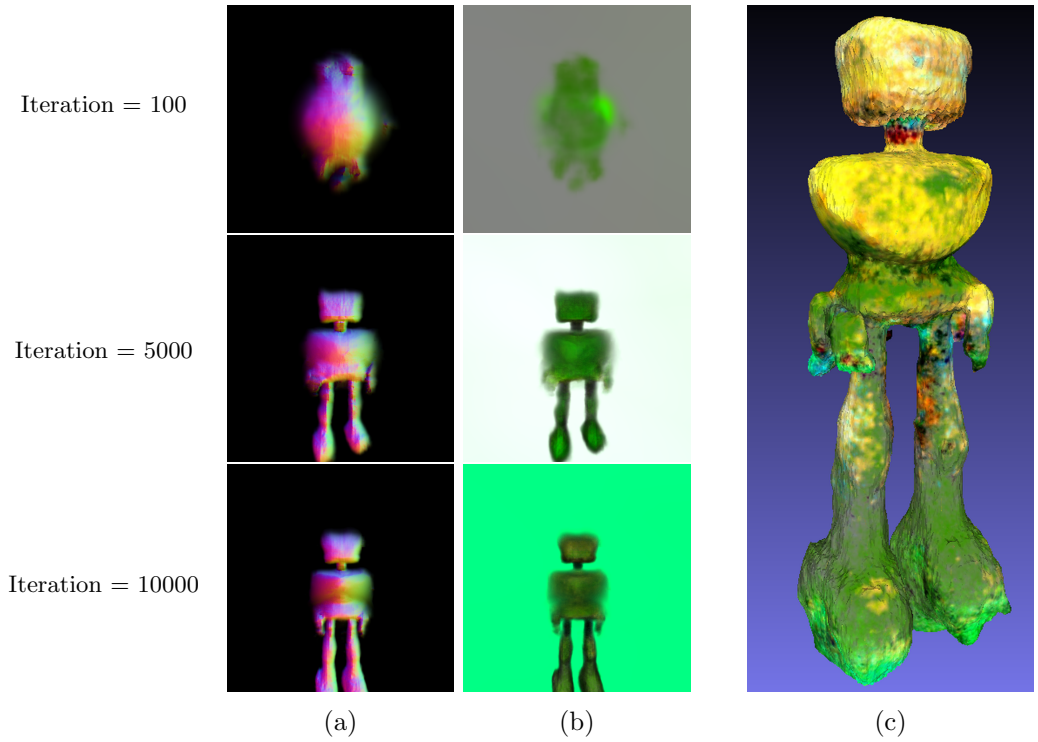


Figure 16: The generation process of Dreamfusion using the prompt "a robot made out of plants". Section (c) shows a snapshot of the final mesh generated.

As seen in Figure **??** 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 [**?**]. 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.

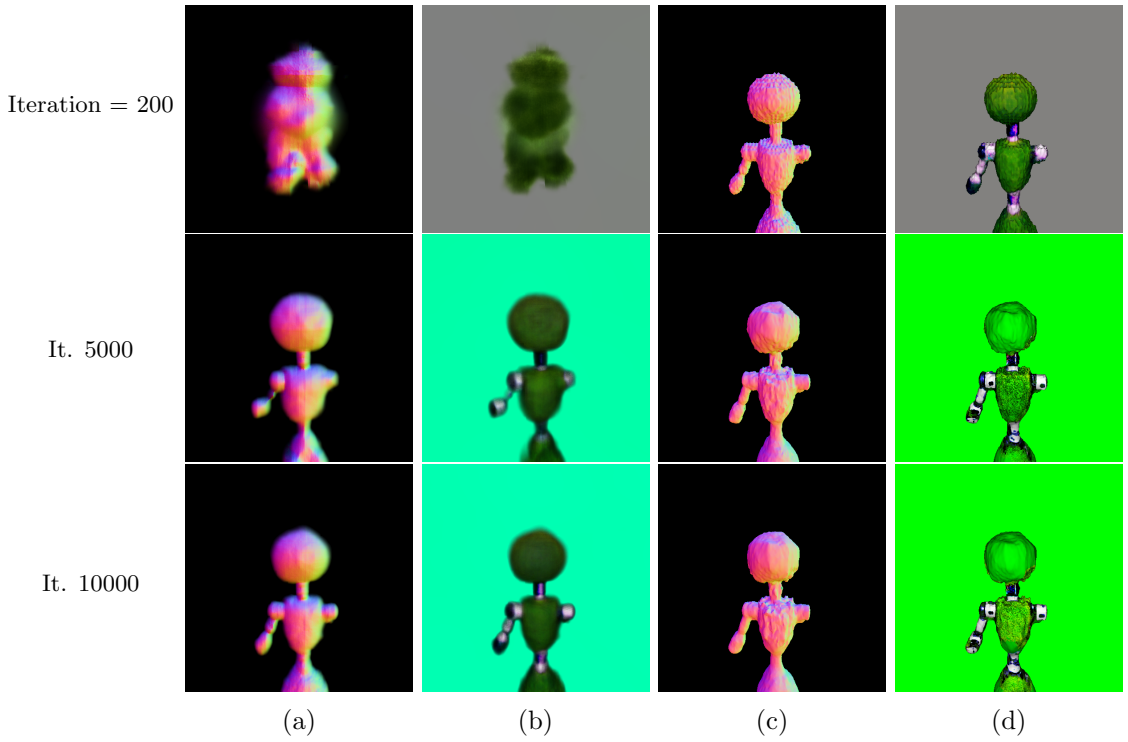


Figure 17: Magic3D generation process from coarse (a, b) to fine (c, d)

The coarse stage can be seen in Figure **??** 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 **??**, 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.

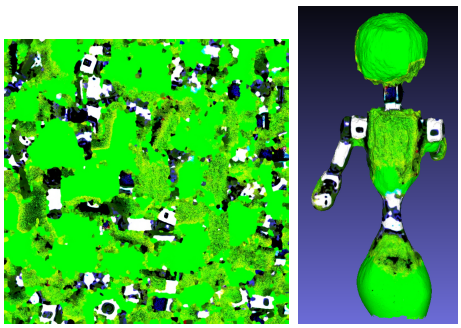


Figure 18: Magic3D also generates an albedo during training. The right side shows the extracted mesh.

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 **??** in the appendix, where the generation process was started with a rough human figure.

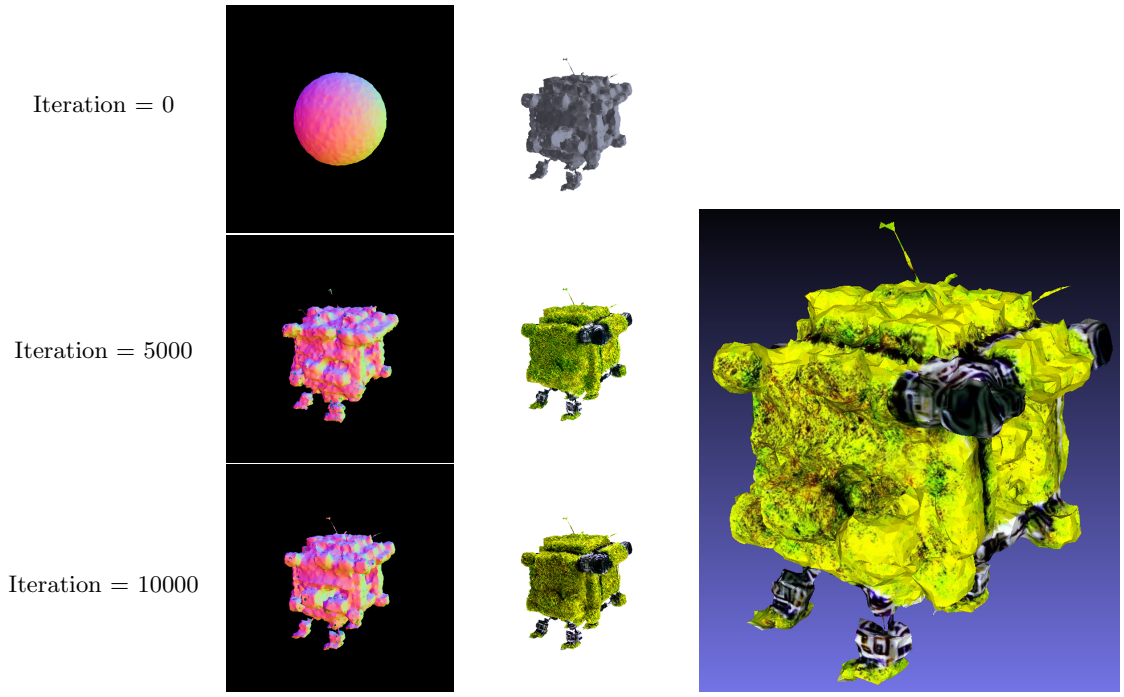


Figure 19: a: Fantasia3D starting with only a perfect square and refining this according to the prompt "a robot made out of plants". In b: only the appearance of the model gets refined. Image c shows the rendered model

Part (a) of Figure **??** 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 **??**.

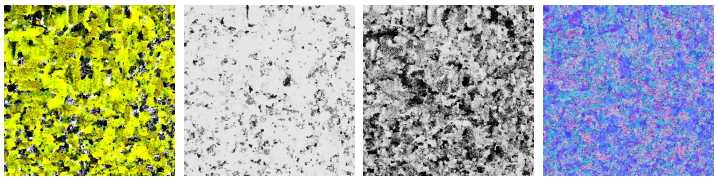


Figure 20: Generated textures from Fantasia3D; from left to right: diffuse, roughness, metallic, and normal.

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 **??** 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.

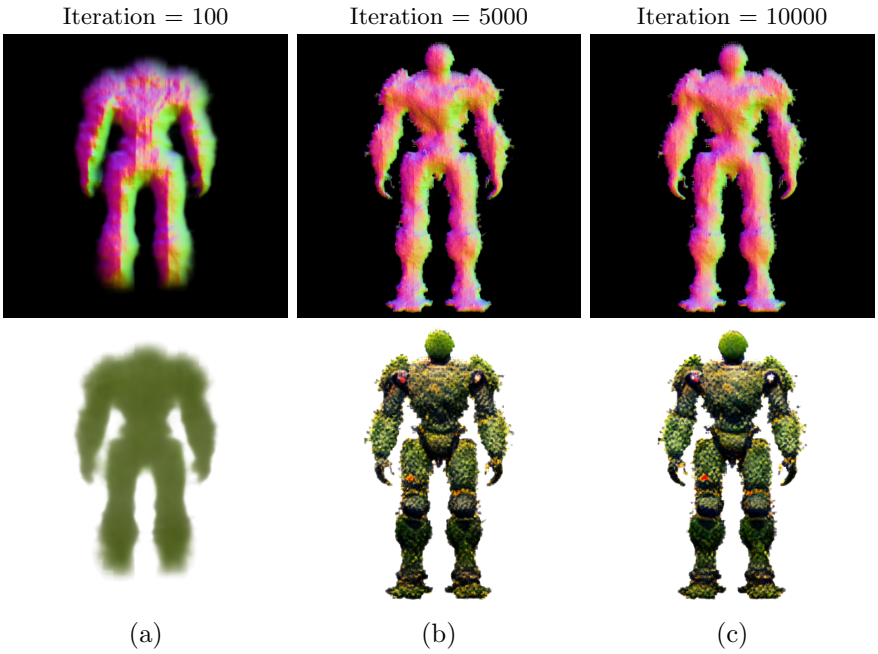


Figure 21: Front view of the coarse stage of Magic123

The front view of the coarse generation process is depicted in Figure **??**. 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 **??**, 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.

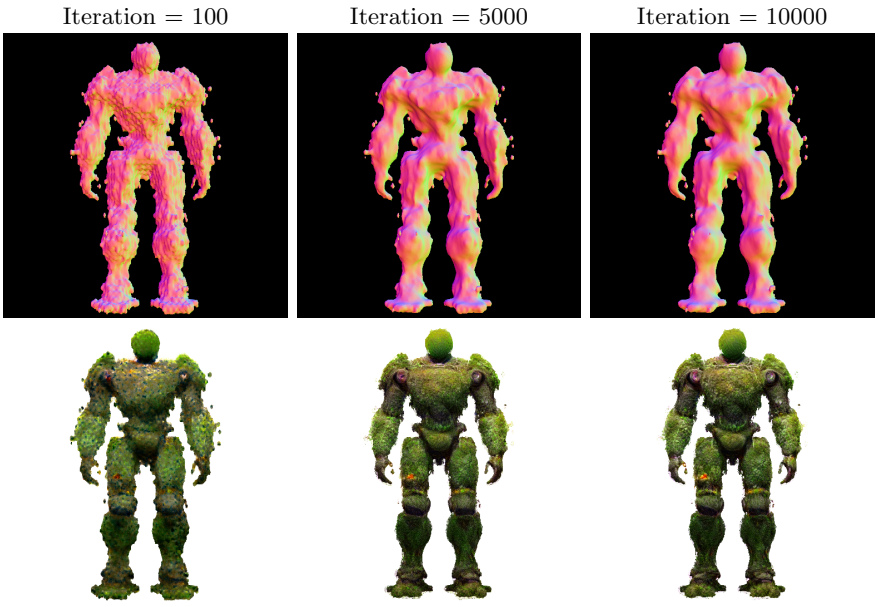


Figure 22: Front view of the refine stage of Magic123

As Magic123 generates multiple views during training, additional perspectives of the object for each iteration can be seen in Figures **??** and **??** 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 **??**, 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 23: 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 **??** part (b). Figure **??** 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 **??**. 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.

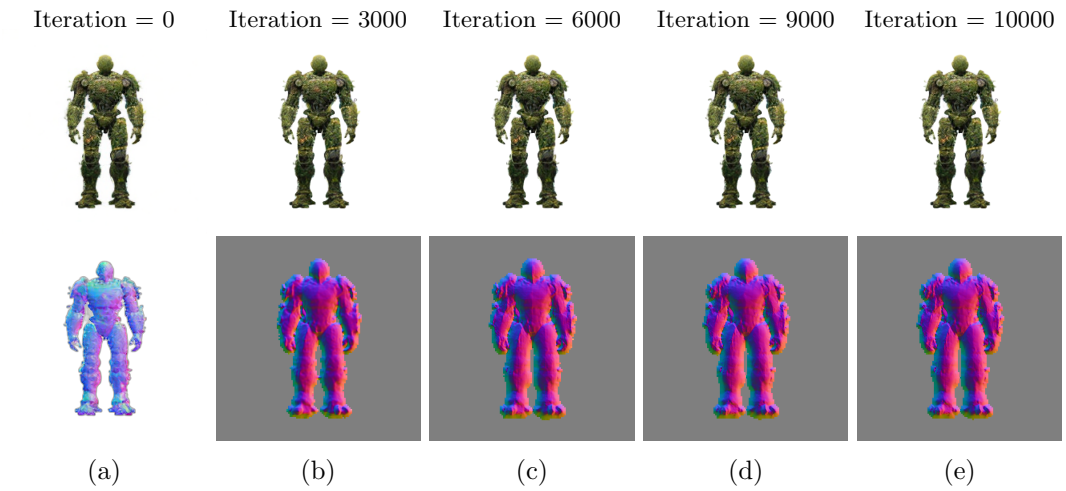


Figure 24: Front view of the Wonder3D generation process. Only minor changes can be seen between initialization and iteration 10000.

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.

In light of these diverse requirements, this analysis of Section **??** evaluates the generated models across multiple dimensions:

1. **Prompt/Result Fidelity:** Investigates how closely each model aligns with the provided prompt or image. Can the intended subject be identified without prior knowledge of the input?

2. **Detail Level:** Examines the intricacy of each model. Are the models finely detailed or do they exhibit a more generalized structure?

3. **Texture Realism:** Assesses the authenticity and quality of the model textures. How 'real' do they look like?

4. **Topology:** Looks into the smoothness or blockiness of the model's surface. Is the topology uniformly smooth or does it exhibit a more fragmented appearance?

5. **Model Integrity:** Considers whether the model is a single, cohesive unit or split into multiple segments.

Additionally, technical aspects of each model are analyzed, such as the time required for rendering, efficiency, and resource consumption. Each method is also subjected to specific tests based on particular requirements. For example, when symmetry is a key

factor, models are evaluated on their ability to produce symmetric outputs. Similarly, when smoothness is crucial, special attention is given to the topology.

The technical metrics for this analysis are derived from the tensorboard and outputs generated during the training process. The percentages and detailed assessments of geometrical features like symmetry, topology, and the presence of holes in the models are calculated using Evaluate3D. This tool, developed as part of this research, leverages the trimesh Python library [**?**] to extract detailed insights into the fundamental geometric properties of each 3D model.

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 **??**.

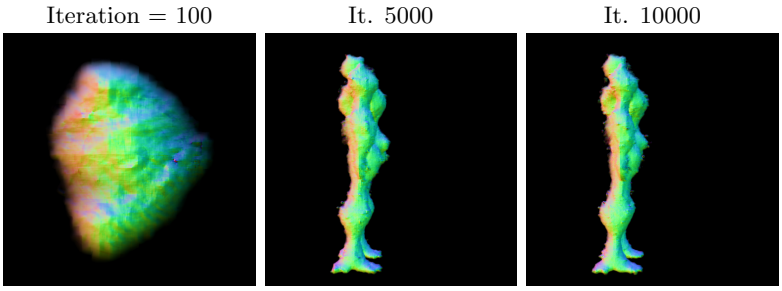


Figure 25: Results obtained using the prompt "a high-quality rendering of a Playmobil firefighter"

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.

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 [**?**] and Gaussian Splatting [**?**].

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 [**?**], 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 [**?**], offers an accessible platform for 3D model generation. This service, operational on a Discord server [**?**], 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 [**?**], 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 **??** 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

The issue of 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 [**??**]. 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 [**?**].

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 [**?**].

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.

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

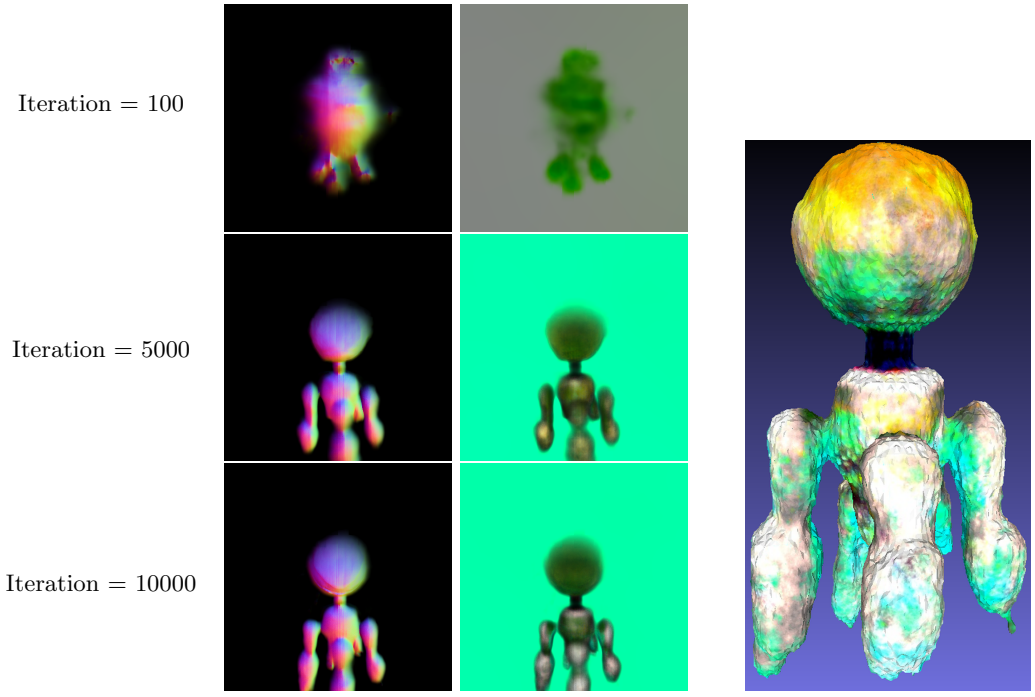


Figure 26: The generation process of DreamFusion using the prompt “a robot made out of plants” a second time.

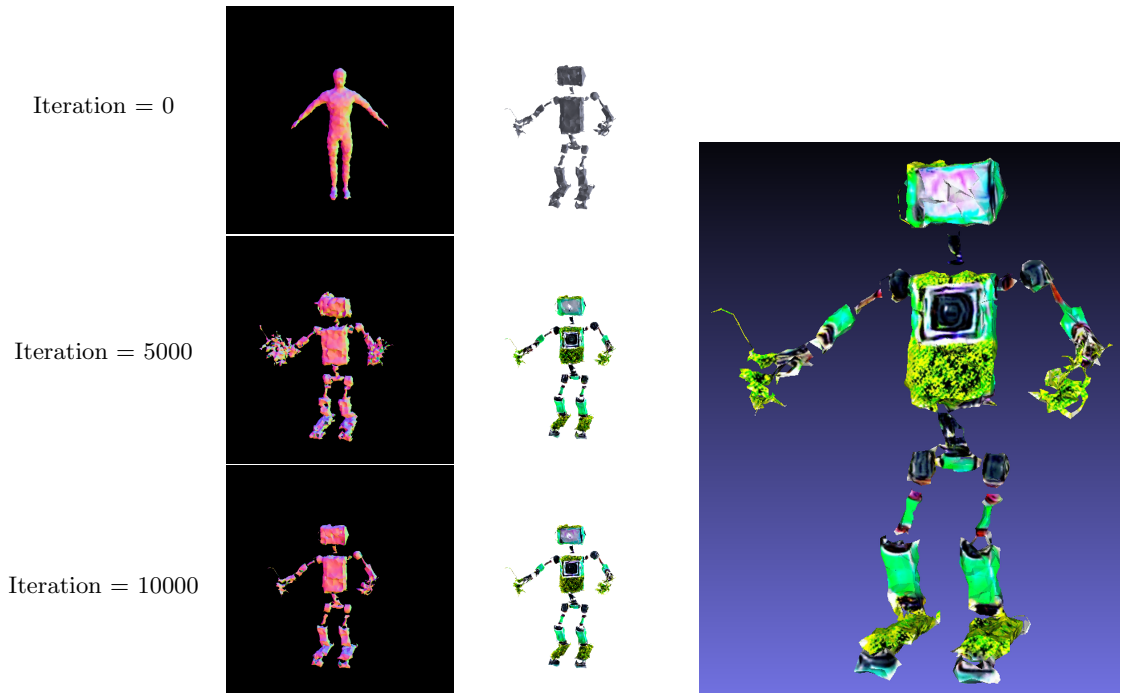


Figure 27: Initializing Fantasia3D with a coarse mesh representing a basic human figure

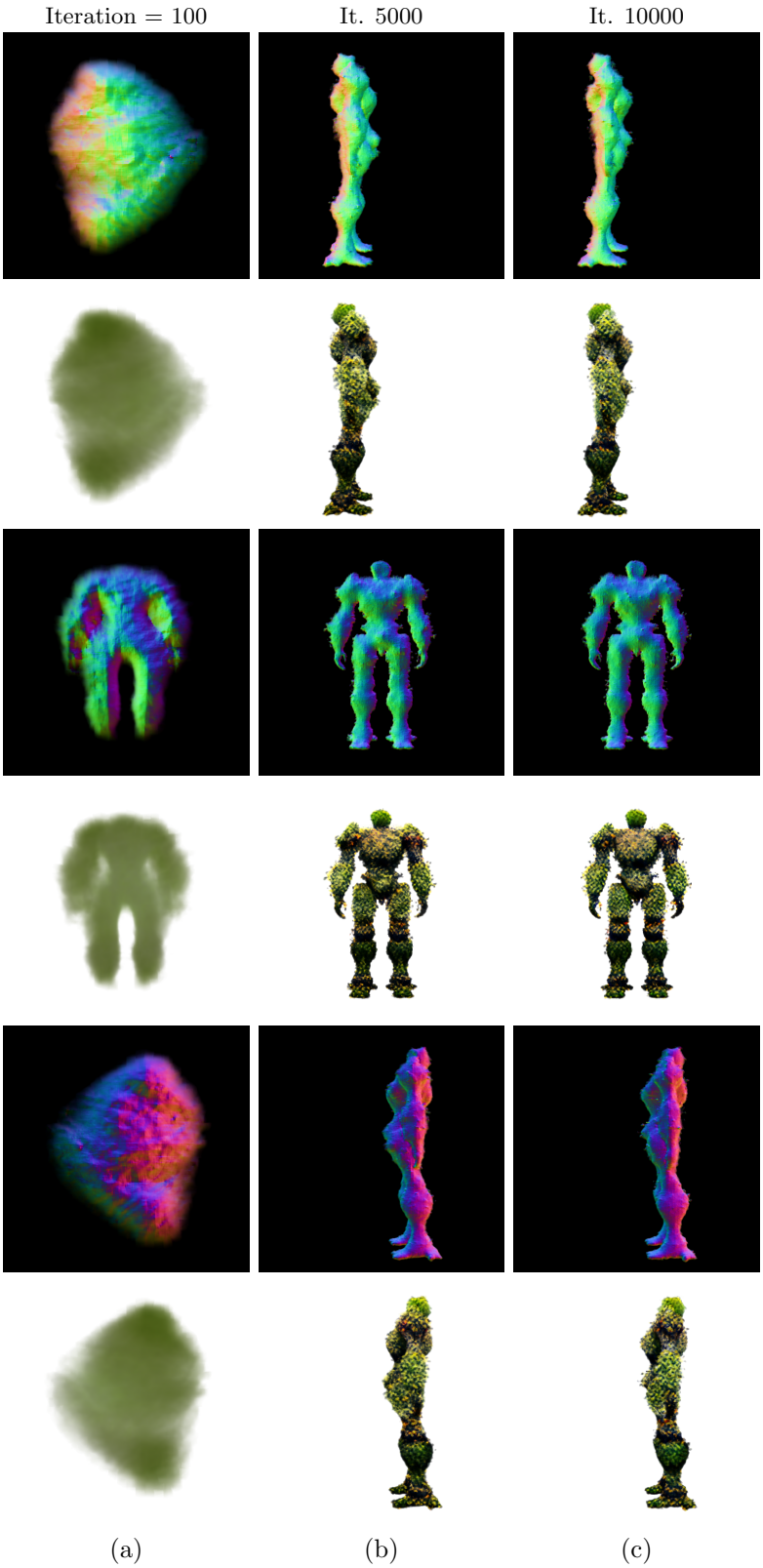


Figure 28: The coarse in Magic123; From top to bottom: right, back and left view

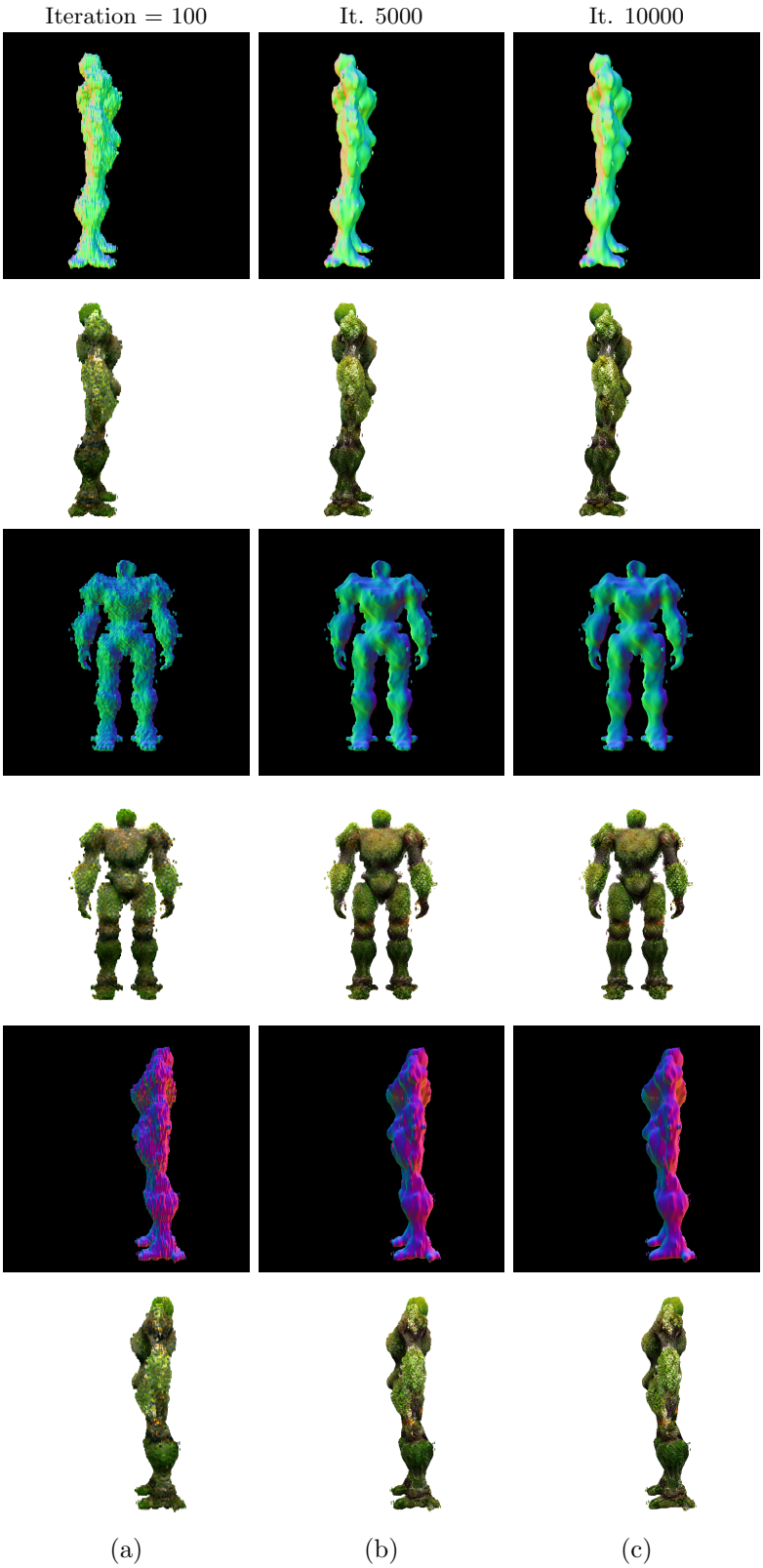
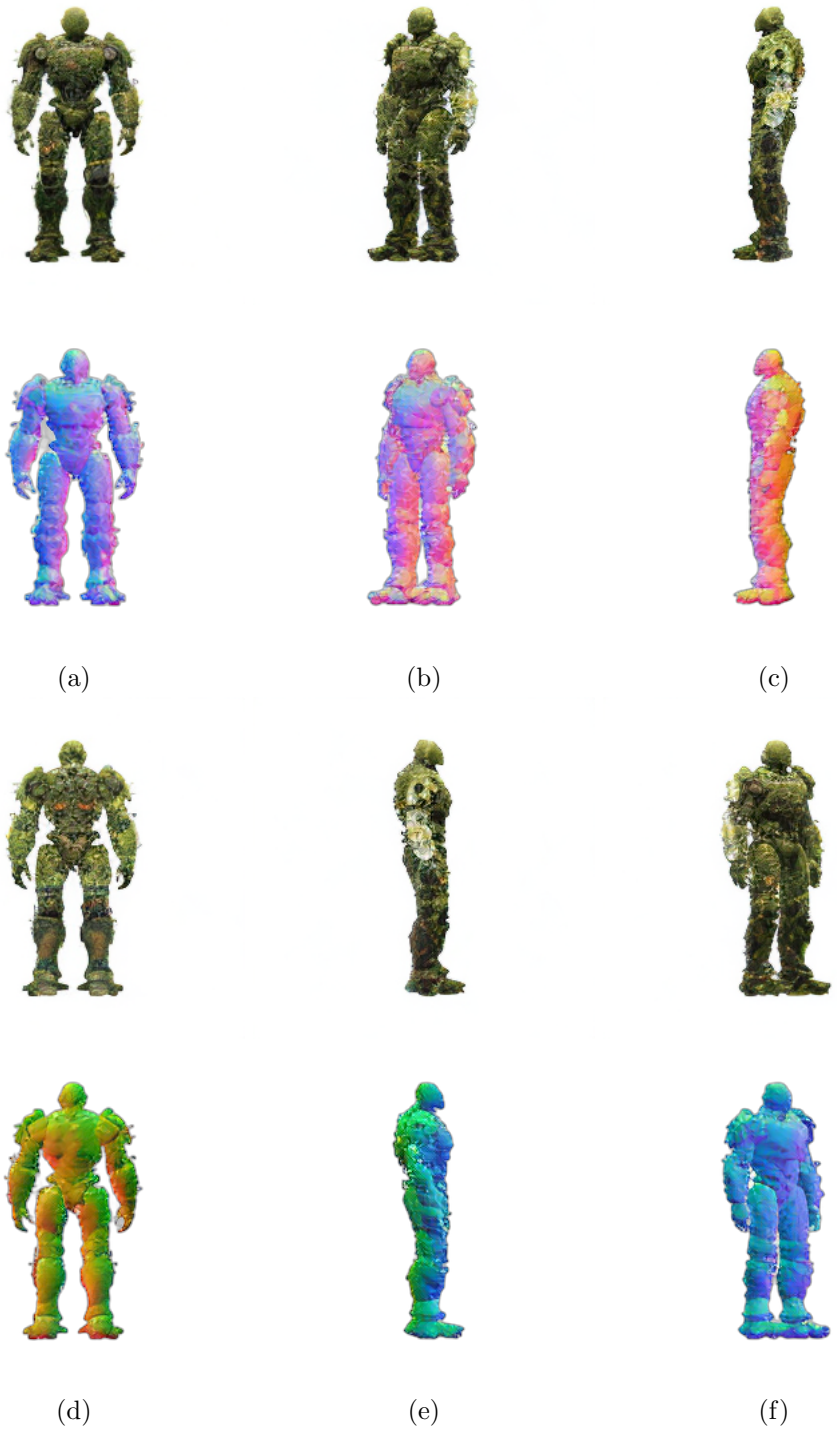


Figure 29: The refine in Magic123; From top to bottom: right, back and left view



(a) (b) (c)

(d) (e) (f)

Figure 30: Wonder3D initialization of multi-view color images and normals; (a) front, (b) front left, (c) left, (d) back, (e) right, (f) front right

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.

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. *arXiv preprint arXiv:2303.13873*.

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*.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

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, MA, 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].

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

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.

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, 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*.

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.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

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

[TODO: Fügen Sie hier die Erklärung zur selbstständigen Bearbeitung gemäß Ihrer Themabestätigung ein]

Datum	Unterschrift