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. In response, researchers and practitioners have leveraged generative AI techniques to develop innovative methods that can automate the process of creating 3D models. These innovations have the potential to not only reshape the way we interact with digital environments, but also to facilitate the simulation, analysis, and visualization of complex real-world phenomena.

The overall goal of this thesis is to provide a comprehensive examination of advances in automated 3D model generation. Efforts are directed at highlighting the techniques and methodologies that underlie this transformative field. Through a careful comparative study, the capabilities of these technologies are evaluated and questions are raised about their potential for creating 3D models that are characterized by both precision and aesthetic appeal. This research contributes to the ongoing development of computer graphics and the broader field of artificial intelligence.

To provide a foundation for this exploration, a comprehensive examination of the fundamentals of generative AI is undertaken. Essential generative models such as Variational Autoencoders (VAEs) [Kingma and Welling, 2022; Rezende et al., 2014], Generative Adversarial Networks (GANs) [Goodfellow et al., 2020] and Diffusion Models [Yang et al., 2022; Ho et al., 2020; Sohl-Dickstein et al., 2015] are explained. Additionally, a breif introduction in Contrastive Language-Image Pre-training (CLIP) [Radford et al., 2021] and various forms of 3D data representation is given. These fundamental concepts provide the necessary foundation for a subsequent in-depth analysis of 3D model generation techniques.

The research also extends to evaluating different approaches to generating 3D models. These approaches include the creation of 3D objects from images, text input, and video sequences, with each approach presenting a unique set of challenges and opportunities. In this thesis, these methods are examined in depth through a comparative study. This investigation is facilitated by well-defined experimental setups, the application of rigorous performance metrics, and the performance of comprehensive results analyses.

In addition, this inquiry addresses future opportunities in the field of 3D modeling. Emerging trends and potential research directions that will redefine the 3D modeling landscape are highlighted. The practical implications of these research findings are carefully considered to highlight their practical application and importance.

//TODO briefly explain test metrics / how evaluated.

Chapter 2

Basics

The 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 generative machine learning algorithms and 3D data representations that form the basis of this field.

The section starts with Variational Autoencoders (VAEs) [Kingma and Welling, 2022; Rezende et al., 2014], which serve as probabilistic frameworks for learning complex data representations. Generative Adversarial Networks (GANs) [Goodfellow et al., 2020] are then discussed, highlighting their unique training dynamics that include both generator and discriminator components. The section further explores the domain of Diffusion Models, particularly Denoising Diffusion Probabilistic Models (DDPMs) [Ho et al., 2020; Sohl-Dickstein et al., 2015], while also briefly covering Stochastic Gradient Methods (SGMs) [Song and Ermon, 2019] and Stochastic Differential Equations (SDEs) [Song et al., 2020, 2021]. These mentioned models, as captured by the "generative learning trilemma" [Xiao et al., 2022], highlight the balance and compromises inherent in their design and application.

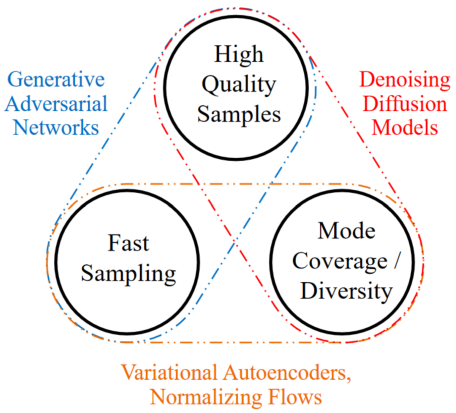

Figure 1: A short characterization of the most important features and drawbacks of the generative models discussed in this section. [Xiao et al., 2022]

The chapter also examines Contrastive Language-Image Pre-training (CLIP) [Radford et al., 2021], showcasing its ability to bridge natural language and visual data, thereby facilitating text-guided 3D model generation. Lastly, various forms of representation of 3D

data are examined, including Meshes, Point-Clouds, and Voxels, constituting the foundational structure of 3D objects in computational environments. Proficiency in these representations is required for comprehending the 3D model generation.

//TODO check Final Diffusion Model

2.1 Variational Autoencoders – VAEs

Generative models, as defined in the seminal work by Diggle and Gratton, fall into two primary categories: prescribed and implicit models. Prescribed models leverage well-formulated, often parametric, mathematical expressions for the probability density function (pdf), thus facilitating easier analytical interpretation of the distributions. Implicit models, on the other hand, synthesize new data samples without an explicit pdf, approximating the data distribution based on the training dataset [Diggle and Gratton, 1984]. Variational Autoencoders (VAEs) are aligned with the prescribed models category, given their reliance on an explicit formulation of the pdf to operate efficiently. This feature makes VAEs suitable for tasks that require not only the generation but also the understanding of complex data distributions [Kingma and Welling, 2022; Rezende et al., 2014; Goodfellow et al., 2016]. In contrast, Generative Adversarial Networks (GANs) are a prime example of the implicit models category [Goodfellow et al., 2020].

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 substantial information after dimensionality reduction. The encoder compresses the input data into a low-dimensional representation, or "code" [Hinton and Salakhutdinov, 2006; Goodfellow et al., 2016], which captures the most relevant features of the input, within a latent space. The decoder then reconstructs the original input from this compressed latent vector, with the error between the output and original data being backpropagated to optimize the model's weights [Goodfellow et al., 2016; Michelucci, 2022]. Although this process balances data compression and information preservation, a drawback remains that if the encoder and decoder are too closely matched, the latent space becomes uncontrollable resulting in a lack of interpretability and regularity [Michelucci, 2022].

Autoencoders, in their essence, aim for approximate rather than perfect replication of the input data, which forces the model to prioritize certain aspects of the input to copy, often learning useful properties of the data [Goodfellow et al., 2016]. This dimensionality reduction proves beneficial in enhancing classification tasks' efficiency by reducing computational costs and memory overhead, and improving information retrieval in low-dimensional spaces [Goodfellow et al., 2016]. However, traditional autoencoders fall short in generating new data due to the unregulated nature of the latent spaces.

The challenge posed by unregulated latent spaces in autoencoders is great when generating new data. Attempting to decode a randomly selected point from this latent space often leads to unusable results because part of the latent space does not match any data point. Simply trying to organize the latent space proves difficult as its structure is affected by the distribution of the source space, the dimensionality of the latent space, and the architecture of the encoder [Michelucci, 2022]. In essence, traditional autoencoders are not designed to generate new data; their main function is to copy and reconstruct the given input [Goodfellow et al., 2016].

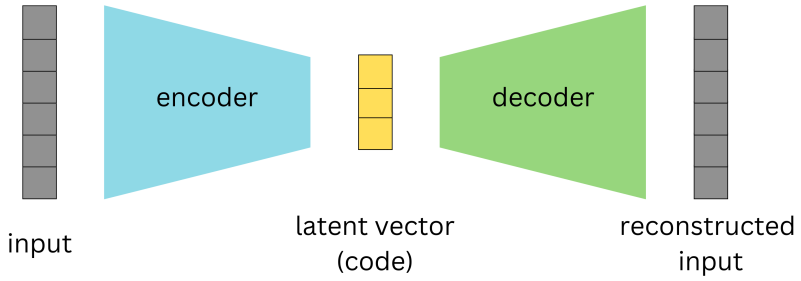

Figure 2: Autoencoder: The encoder reduces the input dimension to a latent vector that captures the most important features. The decoder then uses this vector to reconstruct the input, with training aimed at minimizing reconstruction loss. TODO ADD LOSS https://towardsdatascience.com/difference-between-autoencoder-ae-and-variational-autoencoder-vae-ed7be1c038f2

VAEs tackle the limitations of traditional autoencoders by implementing enhanced regularization of the latent space during training. Although the iterative process largely remains the same, VAEs deviate by encoding a distribution, shaped by the parameters μ and σ, over the latent space instead of a single point [Doersch, 2016]. Then, a point is sampled from this distribution to represent the latent variable z. This sampled point is used to calculate the reconstruction error, which is then backpropagated to update the model's weights.

A major objective of VAEs is to compute the data likelihood, $P(X)$, through strategic sampling of latent variables z. This computation is important because it follows the intuition of maximum likelihood, which states that "if the model is likely to produce training set samples, then it is also likely to produce similar samples, and unlikely to produce dissimilar one" [Doersch, 2016]. Strategic sampling here means that the model uses a new function $Q(z|X)$ that, given a value of X, returns a distribution over latent variables z that is most likely to have produced the data point X [Doersch, 2016]. The core equation for VAEs is expressed as follows:

$$\log P(X) - D_{KL}\left[Q(z|X) \parallel P(z|X)\right] = \mathbb{E}_{z\sim Q}\left[\log P(X|z)\right] - D_{KL}\left[Q(z|X) \parallel P(z)\right]$$

[Doersch, 2016]. This equation first aims to maximize the log likelihood of the data, denoted as $\log P(X)$. A higher log likelihood means that the model is good at recreating data. The KL divergence term D_{KL} acts as an error term, ensuring that $Q(z|X)$ accurately reproduces the posterior distribution $P(z|X)$ [Doersch, 2016]. By minimizing the divergence of P and Q, it is ensured that Q closely resembles P. On the right hand side, the first part determines the expected log-likelihood of the reconstruction data given the latent variables z, under the distribution Q. The KL divergence between the posterior $Q(z|X)$ and the prior distribution $P(z)$, helps to regularize Q. This side of the equation can be optimized using stochastic gradient descent with a suitable choice of Q [Doersch, 2016].

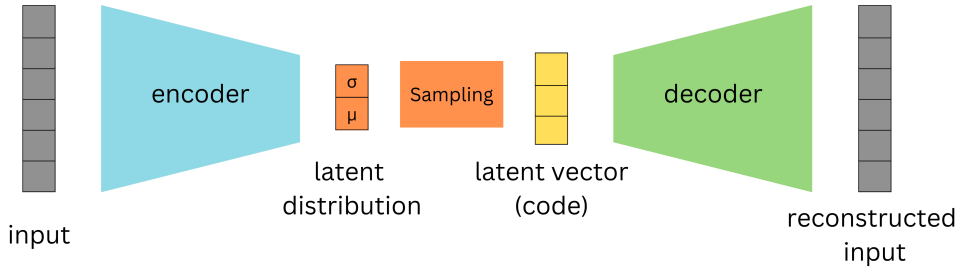

Figure 3: Functionality of a Variational Autoencoder, demonstrating incorporation of the latent distribution – the mean and standard deviation – for enhancing latent space regularization.

Despite their capabilities, VAEs exhibit some limitations. According to Goodfellow et al., the generated samples can often be blurry. The reason for this is not fully understood, but the blurriness observed may be due to their optimization process, which minimizes Kullback-Leibler divergence. This could lead the model to assign high probabilities to "points that occur in the training set, but may also assign high probability to other points [...] which may include blurry images" [Goodfellow et al., 2016]. The Gaussian distribution often used in VAEs for the generative model may also contribute to this effect, as it can ignore minor features in the input data [Goodfellow et al., 2016]. Another issue is that VAEs typically utilize only a small portion of the latent space, which might further compromise the quality of generated images [Goodfellow et al., 2016]. The performance of the model is also sensitive to the choice of priors for the latent space, making hyperparameter tuning an essential aspect of working with VAEs [Kingma and Welling, 2022; Higgins et al., 2017].

2.2 Generative Adversarial Networks – GANs

There are two main approaches in Machine learning methods, supervised learning and unsupervised learning. Supervised learning requires extensive control and a vast amount of labeled data sets, whereas unsupervised learning offers a more straightforward approach by not requiring classified data to learn a model.

In supervised learning, the ML algorithm learns from a labeled data set where each data point is related to a corresponding target or output value. This enables the algorithm to make predictions or classify new, unseen data based on the patterns it has learned from the labeled examples. The model's predictive capabilities are continuously refined by comparing its outputs to the expected outputs from the training dataset. Through this iterative process, adjustments can be made to enhance the model's performance and generate improved outputs. Yet, it is quite expensive and time-consuming to obtain the large amount of required data as it is often labeled manually by human experts. On the other hand, with the use of generative modeling, Unsupervised learning aims to discover patterns or structures within some unlabeled data. It demonstrates to be particularly

useful when labeled data is scarce or unavailable. This concept of unsupervised learning sets the stage for understanding how implicit generative models like GANs address a significant limitation and offer a distinct advantage.

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

The adversarial aspect of GANs arises from the game-like competition between two neural networks: the generator and the discriminator. The generator is responsible for creating fake inputs or samples, which are then passed to the discriminator. The discriminator's role is to differentiate between real samples from the domain set and the fake samples generated by the generator.

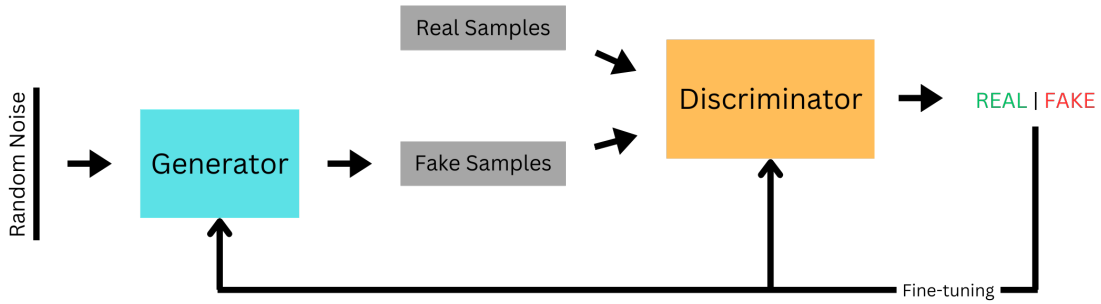

Figure 4: Simplified functionality of a Generative Adversarial Network

Initially, the discriminator is trained on a dataset of unlabeled data, aiming to learn the characteristics and attributes of the desired output. Once it becomes proficient at identifying the genuine objects, it is presented with examples of non-objects and its ability to distinguish these examples is assessed. Subsequently, the generator utilizes random input vectors to generate counterfeit versions of the desired objects. The discriminator then assesses the authenticity of these outputs and shares the result. Based on this feedback, the generator or the discriminator adjust their behavior. This iterative process, facilitated by an iterator, involves creating samples, updating the models, and repeating the cycle. Gradually, the generator becomes highly skilled at producing realistic outputs that the discriminator can no longer distinguish from real ones. This process is referred to as a zero-sum game, where there is always a winner and a loser. The winner remains

unchanged, while the loser updates its model based on the feedback received from the discriminator.

For Images, the Discriminator and the Generator are often implemented as Convolutional Neural Networks (CNNs), which excel at recognizing patterns in images and are commonly used for object identification. GANs are not limited to images, they can also be applied to tasks such as video frame prediction, image enhancement to improve image quality, and encryption [Goodfellow et al., 2020].

GANs pose a substantial challenge in their training process as they are hard to train [Goodfellow et al., 2020]. In addition, Brophy et al. highlight three important problems commonly associated with GANs, among others. These issues, namely non-convergence, diminishing or vanishing gradients, and mode collapse, contribute to the inherent instability experienced during GAN training. Non-convergence refers to the failure of a GAN model to stabilize and reach a state of equilibrium. Instead, it continuously oscillates and fails to converge to a satisfactory solution. As a result, the model does not learn the underlying patterns of the data and can even diverge, leading to poor performance [Brophy et al., 2023]. Diminishing or vanishing gradients occur when the gradients used to update the generator become extremely small or even vanish altogether. This phenomenon is often caused by an overly successful discriminator that becomes too adept at distinguishing real and fake samples. As a result, the generator struggles to learn from the feedback provided by the discriminator, impeding its ability to generate high-quality samples [Brophy et al., 2023]. Mode collapse happens when the generator collapses, meaning it focuses on producing only a limited set of samples or outputs, typically lacking diversity and variety [Salimans et al., 2016]. In such cases, the generator fails to capture the full range of patterns and characteristics present in the training data, resulting in uniform and repetitive samples that do not adequately represent the true distribution [Brophy et al., 2023].

2.3 Diffusion models

The limitations of VAEs and GANs, which were just stated above, have led to the emergence of diffusion models, a method that offer distinct advantages over traditional generative models. Diffusion models operate by progressively perturbing data with noise and then learning to reverse this process to generate new samples.

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

2.3.1 Denoising Diffusion Probabilistic Models

DDPMs employ two Markov chains, a forward chain and a reverse chain, also known as the forward and reverse diffusion processes, seen in Figure 5 and Figure 6 [Sohl-Dickstein et al., 2015].

The forward diffusion process is comparable to latent variable models, sharing some similarities with Variational Autoencoders (VAEs), as focus is lying on a latent feature space of the initial data distribution. They differ in the fact that the forward process in DDPMs “is fixed to a Markov chain that gradually [over a span of T steps] adds Gaussian

noise to the data according to a variance schedule $\beta_1, \ldots, \beta_T$" Ho et al. [2020]. This iterative process continues to add noise "until all structures are lost" [Yang et al., 2022] resulting in an image of pure noise. The introduction of noise aims to gradually steer the data distribution towards a more manageable prior distribution [Yang et al., 2022; Poole et al., 2022]. Mathematically, the forward process can be described in two equations:

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

[Martínez et al., 2023]. This describes the process of adding noise to transform a data point x_{t-1} into a new data point x_t. This transformation is probabilistic and follows a Gaussian distribution [Sohl-Dickstein et al., 2015; Ho et al., 2020]. The mean value of this distribution is slightly adjusted compared to the previous data point by the factor $\sqrt{1-\beta_t}$, which essentially corresponds to the data point x_{t-1} with a certain noise reduction. The parameter β_t controls the amount of noise added, where a larger β_t means more noise [Kingma et al., 2023]. The covariance matrix, denoted $\beta_t I$, implies that each element of the data is independently modified with the same amount of noise, since I represents an identity matrix where all outer diagonal elements are zero [Croitoru et al., 2023].

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

[Martínez et al., 2023]. In the second equation, the focus is on the entire sequence of data points from the original x_0 to x_T, including all intermediate points [Martínez et al., 2023]. This expresses the idea that to understand the probability of the entire noisy trajectory, one can calculate the probability of each step from x_{t-1} to x_t and then multiply these probabilities together. This is due to the Markov property, which states that each step is only dependent on the immediately preceding step [Martínez et al., 2023]. This enables a comprehensive view of the transition probabilities over the entire process of noise induction, step by step.

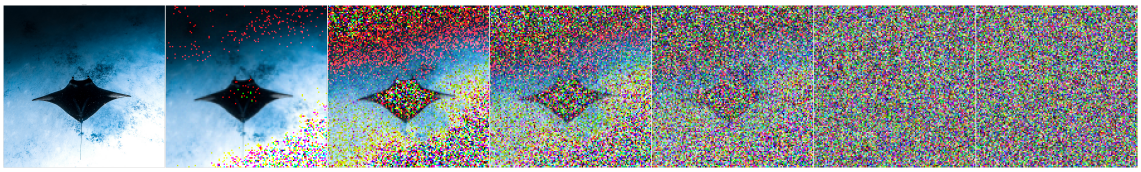

Figure 5: Forward process adding noise to an image

The reverse chain is trained to approximate the inverse of the forward process using a deep neural network parameterized with Θ, effectively removing the noise added by the forward chain [Sohl-Dickstein et al., 2015; Yang et al., 2022]. The function $p_\theta(x_{t-1}|x_t)$ is a probability distribution that estimates how to reverse the diffusion process. Given a data point x_t, it attempts to predict the previous data point x_{t-1} before noise was added. It is modeled as a normal distribution with a mean $\mu_\theta(x_t, t)$ and covariance $\Sigma_\theta(x_t, t)$, both of which are functions parameterized by θ [Yang et al., 2022].

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

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

[Martínez et al., 2023]. The latter function $p_\theta(x_{0:T})$ represents the probability of the entire sequence of data points from x_0 through x_T under the reverse process modeled by the neural network. It starts with an estimate of the final data point $p_\theta(x_T)$ and works backwards through the sequence, multiplying the conditional probabilities $p_\theta(x_{t-1}|x_t)$ for each step [Ho et al., 2020; Martínez et al., 2023]. This function essentially provides a framework for reconstructing the original data from the noisy data by successively removing noise at each step, based on the learned parameters θ [Yang et al., 2022].

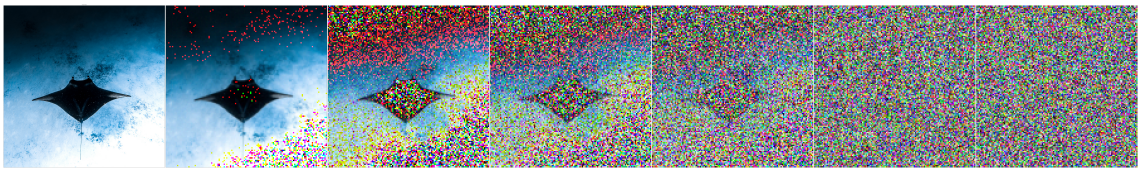

Figure 6: Forward process adding noise to an image

The incremental introduction of the forward and backward diffusion processes offers an advantage as "estimating small perturbations is more tractable than explicitly describing the full distribution with a single, non-analytically-normalizable, potential function" [Sohl-Dickstein et al., 2015]. According to [Ho et al., 2020], the neural network in the reverse process can be trained to predict one of three possibilities: the mean value of the noise at each time step, the original image itself, or the noise of the image [Ho et al., 2020]. As previously mentioned, the second approach is not as advantageous. Hence, the research focuses on the first and last possibilities, which are essentially identical but parameterized differently. Predicting the image noise allows for straightforward subtraction of the noise from the image, resulting in a less noisy version. By employing this method iteratively, it becomes possible to completely learn an image from noise.

Nevertheless, DDPMs also have their drawbacks with the most severe being the time requirement to generate new samples [Xiao et al., 2022]. "This is caused by the fact that a Markov process has to be simulated at each generation step, which greatly slows down the process" [Martínez et al., 2023].

2.3.2 Score-Based Generative Models

SGMs [Song and Ermon, 2019] take a unique approach to generative modeling by prioritizing the learning of a score function that plays a central role in guiding the generative process. This function aims to capture the Stein score [Liu et al., 2016], which is essentially "the gradient of the log-density function at the input data point" [Song and Ermon, 2019]. The score can be viewed as a vector field that indicates the direction that increases the data density the most [Song and Ermon, 2019]. This reveals how slight adjustments within the data can affect the overall distribution, informing the model on how to proceed with sample generation in a way that matches the actual data structure.

To train a neural network for SGMs, Song and Ermon [2019] employ two technique, namely Score matching and Langevin dynamics [Hyvärinen and Dayan, 2005]. Score

matching allows to train a neural network, referred to as a score network $s_\theta(x)$, to predict the gradient of the logarithm of the probability density of the data $\nabla_x \log p_{\text{data}}(x)$ directly without the need to first construct a model that can estimate the probability density $p_{\text{data}}(x)$ itself [Song and Ermon, 2019]. The aim of this process is to minimize the difference between the predictions of the score network and the true gradient of the log-likelihood, which under certain conditions ensures that the trained network approximates the true score "almost surely" [Song and Ermon, 2019]. The term $\frac{1}{2}\|s_\theta(x)\|_2^2$ is part of the objective function that needs to be minimized and is used to regularize the scores predicted by the neural network while the term $tr(\nabla_x s_\theta(x))$, involving the trace of the Jacobian [Song and Ermon, 2019] of the score function, further refines the objective function by capturing the divergence of the score vector field.

$$\mathbb{E}_{p_{\text{data}}(x)}\left[\text{tr}(\nabla_x s_\theta(x)) + \frac{1}{2}\|s_\theta(x)\|_2^2\right]$$

[Song and Ermon, 2019] However, scaling score matching to deep networks and high-dimensional data can be challenging due to the high computational effort required to compute certain matrix operations (such as the trace of the Jacobian matrix) [Song and Ermon, 2019]. Denoising score matching and Sliced score matching could overcome these scalability issues for large applications [Song and Ermon, 2019].

In order to generate Samples, Langevin dynamics [Roberts and Tweedie, 1996] is utilized for sampling from a probability distribution by employing the score function, $\nabla_x \log p(x)$ [Song and Ermon, 2019]. The process begins with an initial guess $\tilde{x}_0$ from a prior distribution $\pi(x)$, and iteratively updates this guess according to the rule:

$$\tilde{x}_t = \tilde{x}_{t-1} + \frac{\epsilon}{2}\nabla_x \log p(\tilde{x}_{t-1}) + \sqrt{\epsilon} z_t,$$

[Song and Ermon, 2019] where z_t follows a standard normal distribution and ϵ is a small step size. Theoretically, as ϵ approaches zero and the number of iterations T becomes very large, the distribution of $\tilde{x}_T$ will converge to $p(x)$ [Song and Ermon, 2019]. The score network $s_\theta(x)$ is trained to closely estimate $\nabla_x \log p_{\text{data}}(x)$, thus allowing for the generation of new samples that approximate the desired data distribution through Langevin dynamics [Song and Ermon, 2019].

In the specific context of score-based generative modeling, there's a notable challenge to overcome. The estimated score function may not be entirely accurate in regions of the data space where there's minimal or no training data available [Song and Ermon, 2019]. In such cases, Langevin Dynamics might not converge correctly, resulting in complications during the sample generation process. To address this issue, Song and Ermon [2019] suggest "to perturb the data with random Gaussian noise of various magnitudes", while simultaneously estimate the score functions for these noise-altered data distributions. This approach is called Noise Conditional Score Networks (NCSNs) and ensures that the resulting data distribution doesn't condense into a lower-dimensional structure [Song and Ermon, 2019].

2.3.3 Stochastic Differential Equations

Song et al. [2020] aim to combine both DDPMs and SGMs (NCSNs) using Stochastic Differential Equations (SDEs) to perturb data across an "infinite spectrum of noise scales" [Song et al., 2020]. These SDEs are further used for sample generation [Yang et al., 2022].

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 done through the following SDE

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

[Yang et al., 2022] where the data evolves as noise intensity increases over time. The process is governed by two coefficients: a drift coefficient $f(x,t)$, and a diffusion coefficient $g(t)$, which scales the random noise introduced by Brownian motion dw (Wiener process) [Song et al., 2020]. This Brownian motion represents the random movement of particles in a fluid as they collide with fast-moving molecules in the fluid.

"A remarkable result from Anderson [Anderson, 1982] states that the reverse of a diffusion process is also a diffusion process, running backwards in time and given by the [following] reverse-time SDE:" [Song et al., 2020]

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

This equation describes the process of recovering data from noise by moving backward in time. Knowledge of the score function $\nabla_x \log p_t(x)$ at each time enables this reverse process, allowing for the generation of data samples from the original distribution using numerical techniques [Song et al., 2020].

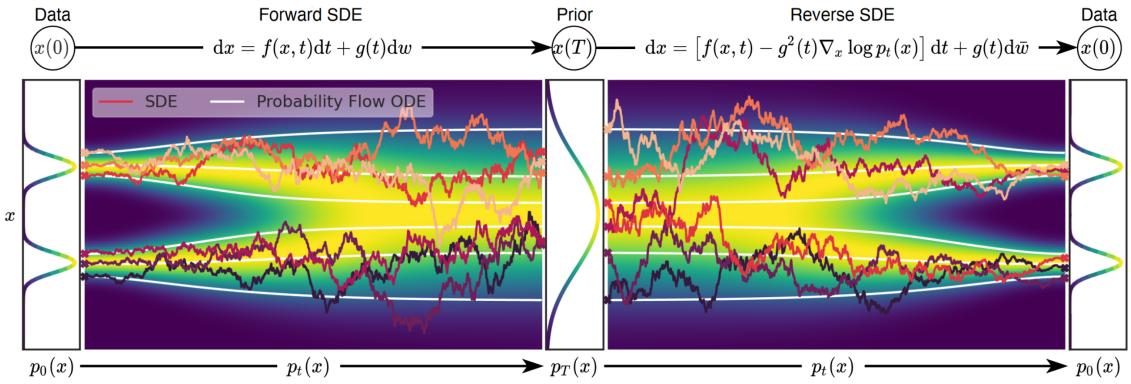

Figure 7: Summary of the score-based generative modeling through SDEs [Song et al., 2020]

knowledge of the scores at each time requires the estimation of the scores for an SDE which involves training a model to approximate the gradient of the log probability of data at various noise levels [Song et al., 2020]. The training objective is given by:

$$\theta^* = \arg\min_\theta \mathbb{E}_t \left\{ \lambda(t) \mathbb{E}_{x_0} \mathbb{E}_{x_t|x_0} \left\| s_\theta(x_t,t) - \nabla_{x_t} \log p_{0t}(x_t|x_0) \right\|_2^2 \right\}$$

[Song et al., 2020] The equation presents a method to find optimal model parameters θ^* that minimize the expected discrepancy between the scores estimated by the model and the actual data transition scores over time, which are influenced by the noise [Song et al., 2020].The expectations are taken over time and modulated by a time-varying weighting function $\lambda(t)$. "With sufficient data and model capacity, score matching ensures that the

optimal solution for the above equation, denoted by $s_\theta * (x, t)$ equals $\nabla_{x_(t)} \log p_t(x)$ for almost all x and t" [Song et al., 2020]. 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 [Yang et al., 2022].

2.4 Contrastive Language-Image Pre-training - CLIP

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

Unlike traditional computer vision models that rely solely on annotated image datasets, CLIP leverages 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 [Radford et al., 2021]. It consists of two main components: an image encoder and a language encoder. The image encoder processes images using a modified version of ResNet50 [He et al., 2016] or as a second approach was build upon the Vision Transformer (ViT) [Dosovitskiy et al., 2021], while the language encoder uses another modified transformer-based model to process textual descriptions [Vaswani et al., 2023]. 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. Furthermore, CLIP has been shown to be very effective on a variety of tasks. It outperforms state-of-the-art models on the ImageNet classification task, and it achieves state-of-the-art results on the Visual Genome dataset for object detection and question answering [Radford et al., 2021].

However, there exist several limitations to CLIP. "The performance of zero-shot CLIP is often just competitive with the supervised baseline of a linear classifier on ResNet-50 features" [Radford et al., 2021]. This means that CLIP is not significantly better than a model that is trained on labeled data for the specific task at hand. In addition, the authors estimate that achieving SOTA performance across their evaluation suite would require significantly more computational resources, approximately a 1000-fold boost in computational power. Current hardware capabilities cannot accommodate such demands, emphasizing the requirement for advancements in hardware technology to effectively train zero-shot CLIP models.

2.5 Representation forms of 3D Data

2.5.1 Meshes

//TODO

2.5.2 Point-Clouds

//TODO

2.5.3 Voxels

//TODO

Chapter 3

Models

There exist several different approaches to generate 3D Models from a 2D template. In this chapter I will provide detailed insights in some of these approaches.

3.1 Neural Radiance Fields – NeRFs

Neural Radiance Fields (NeRFs), as introduced by Mildenhall et al., offer a groundbreaking approach to representing 3D scenes from a limited set of 2D images [Mildenhall et al., 2020]. Unlike conventional 3D reconstruction methods, which often have problems capturing complex geometries and variable lighting conditions, NeRFs use a volumetric representation to accurately capture both the spatial and angular distribution of light. This is achieved by modeling the volumetric scene as a continuous 5D function, accepting 3D spatial coordinates and viewing direction as inputs, and generating the color and opacity at a given point in the scene as outputs. This fundamental change in representation allows NeRFs to address challenges common to traditional techniques and provide a more robust framework for reconstructing and rendering 3D scenes [Mildenhall et al., 2020].

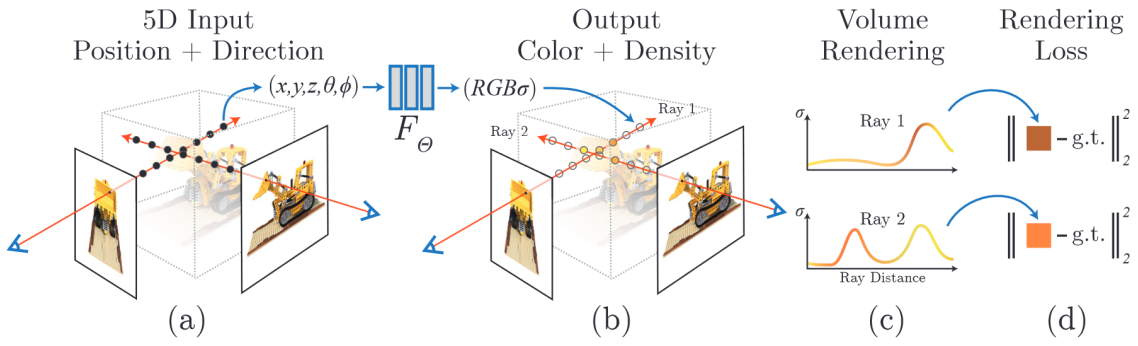

Figure 8: Neuronal Radiance Field. Figure taken From Mildenhall Mildenhall et al. [2020] – Figure 2

Classic deep learning methods often require a comprehensive dataset comprising various scenes and their representations. NeRFs, on the other hand, are trained to specialize in a single, unique scene [Mildenhall et al., 2020]. The underlying neural network consists of Fully Connected Layers (FCs) with ReLU activations, specifically designed to encode the

volumetric details of that particular scene, effectively creating a dedicated neural network for each scene [Mildenhall et al., 2020]. The neural network accepts two types of input: a position in a given coordinate system, often expressed as a 3D vector x, y, z, and a viewing direction represented by two angles θ, ϕ. To better understand the concept of the latter two inputs, 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. These angles help the neural network understand from which direction the beam is being cast to a point in 3D space. The network's output consists of the color c and density σ at that particular location [Mildenhall et al., 2020]. Queries to the neural network can be made by projecting a ray through the scene and asking for color and density information at various points along the ray. This unique querying process enables capturing complex visual phenomena like lighting, reflections, and transparency, which are often challenging to model with traditional 3D reconstruction techniques. For each pixel in a desired image frame, the neural network is queried at multiple points along a ray projected through the scene to produce a curve representing the density of objects along the ray's path, as seen in part (c) of Figure 8. The point at which this density curve rises significantly usually corresponds to an object in the scene, and the color at this point is what is rendered for that particular pixel. These density and color curves can be visualized using graphs to illustrate how density and color vary along the ray's path. For instance, if a ray intersects a tree, the density would increase and the color output would likely be green, representing the tree's leaves [Mildenhall et al., 2020].

The network can also handle multi-view consistency, predicting color as a function of both location and viewing direction, while density is predicted solely based on location. This distinction is grounded in the idea that density, or transparency, is invariant to the direction from which an object is viewed [Hu et al., 2023]. Therefore, to generate the color, the model first computes the density and a hidden representation based on the location. This hidden representation is then concatenated with the viewing direction and passed through another set of layers to produce the color.

An integral aspect of setting up NeRFs involves solving the problem of identifying the camera's position and direction for each input image. Methods like Structure-from-Motion (SfM) and Simultaneous Localization and Mapping (SLAM) can address this issue [Wei et al., 2021]. Once these parameters are identified, new views can be synthesized by querying the neural network for color and density information along rays projected through the scene [Gerats et al., 2023]. Another challenge arises during training due to the absence of explicit density information. Nevertheless, the training process can be managed by minimizing the loss between predicted and observed values from input images. The model employs backpropagation, as its entire pipeline for rendering images, which includes casting rays, sampling along these rays, and integrating the sampled values to compute a pixel's color, is differentiable [Yariv et al., 2020].

In Neural Radiance Fields (NeRF), volume rendering with radiance fields employs the formula

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

This formula [Mildenhall et al., 2020] facilitates the rendering of a 3D scene by cast-

ing rays and aggregating their properties from a near boundary (t_n) to a far boundary (t_f). In simpler terms, as a ray traverses through the scene, this formula accumulates color information based on the density and transparency of objects along the ray's path. Specifically, $T(t)$ quantifies the transparency of the scene at a particular point along the ray, denoting how much light penetrates through, while $\sigma(r(t))$ denotes the density of objects at a particular point along the ray, with higher density implying more matter at that point. The term $c(r(t), d)$ denotes the color at a particular point along the ray, given the viewing direction d. This calculation determines whether the ray continues through empty space or ends when it hits an object, with color adjustments made at this endpoint [Mildenhall et al., 2020].

Although naive NeRF models may not provide photorealistic results due to the lack of detail, several optimizations have been introduced to improve their performance. One notable improvement is the use of positional encoding techniques that deterministically map 3D coordinates and view directions to a higher dimensional space. This is achieved by using high-frequency features before inputting them to the multilayer perceptron (MLP), which helps optimize the neural radiation fields to better represent high-frequency scene content [Mildenhall et al., 2020]. Hierarchical volume sampling is another important 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, the refined or "fine" network is guided to perform a more detailed sampling. This hierarchical approach is critical because dense sampling is very computationally expensive. A two-tier system therefore helps manage computational resources while allowing detailed sampling along rays to obtain the density and color of the sampled points [Arandjelović and Zisserman, 2021].

The experimental results highlight several notable strengths and limitations of the Neural Radiance Fields (NeRF) approach. One of its key advantages is the model's ability to represent fine-grained structures. Another advantage of NeRF is its memory efficiency. For example, rendering a single scene with NeRF requires only about five megabytes of memory, which is in stark contrast to voxel grid renderings that require over 15 gigabytes for a comparable scene [Mildenhall et al., 2020]. This mismatch in memory requirements underscores NeRF's superior efficiency in terms of data storage and transfer, and represents a compelling advantage over traditional 3D rendering techniques [Mildenhall et al., 2020]. Remarkably, the memory requirement of the rendered scene is even smaller than that of the input images, making the model extremely efficient in data storage and transfer [Mildenhall et al., 2020]. 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 to converge, assuming the use of a single NVIDIA V100 GPU [Mildenhall et al., 2020]. This is equivalent to a training period of about one to two days. While this does not require a data center, it does require a significant amount of time, making NeRF less suitable for scenarios that require rapid implementation. In addition, NeRF rendering is prone to sampling and aliasing issues that can lead to significant artifacts in the synthesized images [Rabby and Zhang, 2023]. These artifacts arise from the limited sampling of the radiation field, resulting in inaccurate reconstruction of certain features, especially in scenes containing sharp edges or textures [Rabby and Zhang, 2023]. The slow training speed and rendering, especially at high resolutions, further exacerbate the problems, making NeRF impractical for real-time

applications or for generating animations that span multiple frames [Rabby and Zhang, 2023].

One of the notable strengths of NeRFs lies in their capability for view-dependent appearance modeling. The system considers how the appearance of an object or scene varies with the direction from which it is viewed, offering a more nuanced and realistic visual impression. This view-dependent feature of NeRFs enables the synthesis of images that accurately reflect how differently the same scene or object can appear when viewed from different angles. In addition, NeRFs offer the ability to capture more complicated geometries and occlusions. The method is capable of producing detailed depth maps that are often difficult to obtain using traditional techniques, especially for complex scenes. These depth maps are versatile enough to support mixed reality applications and allow seamless integration of virtual objects into real-world environments. The system also facilitates the conversion of Neural Radiance Fields into 3D mesh structures using marching cubes, further increasing their usefulness. Another advantage of using NeRFs is their ability to process real-world objects captured through inward-looking 360-degree views. Unlike many other methods, NeRFs do not require background isolation or masking, making them suitable for a wide range of applications. They can synthesize any intermediate view between captured images, providing a complete and flexible representation of the object or scene. https://www.matthewtancik.com/nerf

3.2 3D from Image

//TODO

3.3 3D from Text input

//TODO

3.3.1 Dreamfields

Dreamfields present a novel approach to generating three-dimensional objects guided by textual input as introduced by Jain et al.. Utilizing a continuous volumetric representation, Dreamfields are capable of producing high-quality, intricate 3D objects that not only adhere to the specified textual descriptions but also exhibit realistic geometry and appearance. This technology leverages pre-trained image and text encoders, providing a zero-shot learning capability that does not require paired text and object data for training [Jain et al., 2022].

The methodology in Dreamfields hinges on learning a continuous volumetric representation, termed a Dream Field, for both the geometry and appearance of a 3D object. The aim is to optimize this Dream Field to be consistent with a given textual description, extending the ideas from previous work on visualizing preferred inputs and features of neural networks by optimizing in image space [Jain et al., 2022]. The Dream Field is modeled as a neural network, $f : \mathbb{R}^3 \to \mathbb{R}^4$, mapping a 3D coordinate (x, y, z) toa 4D vector (r, g, b, σ) that represents the color and density at that location. The model then employs raymarching to visualize this continuous field, a computational technique that can be understood as moving a camera through the 3D space to capture the object from

various angles. Mathematically, this is achieved by integrating the Dream Field along a ray $R(t)$ emanating from the camera. Essentially, this is like calculating the average color and light intensity the camera "sees" as it looks along this line. The rendered color C and opacity, or density, α are computed using the formulas [Jain et al., 2022]:

$$C = \int_0^T \alpha(t) \cdot c(t) \cdot \exp\left(-\int_0^t \alpha(s)\, ds\right) dt$$

$$\alpha = \int_0^T \alpha(t) \cdot \exp\left(-\int_0^t \alpha(s)\, ds\right) dt$$

Here, $c(t)$ and $\alpha(t)$ are the color and density at point t along the ray, T is the total length of the ray, and exp is the exponential function, used to model how light fades or attenuates as it moves through the object.

The optimization aims to align the rendered image with a given textual description. This phase focuses on minimizing an objective function $\mathcal{L}$, which essentially quantifies the difference between the generated object and the textual description [Jain et al., 2022].

$$\mathcal{L} = -\frac{E_{\text{img}}(I) \cdot E_{\text{text}}(T)}{\|E_{\text{img}}(I)\| \, \|E_{\text{text}}(T)\|}$$

Here, $E_{\text{img}}(I)$ and $E_{\text{text}}(T)$ are the embeddings of the image and text, respectively, converted into a common numerical language by pre-trained encoders. $\mathcal{L}$ is minimized when these embeddings are most similar, measured by the cosine of the angle between them (cosine similarity). In summary, Dreamfields integrates these mathematical principles—continuous mapping through neural networks, raymarching for rendering, and optimization based on similarity measures—to fine-tune the geometry and appearance of a 3D object so that it aligns closely with the given textual description [Jain et al., 2022].

Dreamfields find applications in numerous domains where 3D object generation is crucial. These include but are not limited to computer-aided design, virtual reality, and augmented reality. Their zero-shot learning capability opens doors for creating complex objects without the need for extensive training data, making them particularly useful in scenarios where paired text-object data are scarce or unavailable [Jain et al., 2022].

Dreamfields' approach to generating 3D objects presents a unique blend of advantages and limitations that contribute to its versatility and potential areas for improvement. Among its most notable advantages is the zero-shot learning capability, allowing the model to generate objects based solely on textual descriptions without the need for paired text-object training data. This attribute is particularly advantageous in scenarios where collecting such paired data is either impractical or resource-intensive [Jain et al., 2022]. Another significant strength lies in its use of a continuous volumetric representation, which enables the creation of intricate and nuanced geometries. Unlike traditional methods that rely on discrete representations like voxel grids, Dreamfields can generate highly detailed and realistic 3D models [Jain et al., 2022]. Adding to its merits is the unified treatment of both geometry and appearance in a single representation. Traditional approaches often separate these two aspects, which can result in inconsistencies in the final model [Jain et al., 2022]. Dreamfields elegantly sidestep this issue, producing 3D objects that are both geometrically and aesthetically coherent. Moreover, the model's optimization process offers a high degree of flexibility, capable of adapting to a wide variety of textual inputs. This makes it a versatile tool for generating a diverse range of objects

across different domains. However, Dreamfields is not without its limitations. One of the primary challenges is the computational intensity associated with its continuous representation and optimization process [Jain et al., 2022]. The high computational requirements could constrain its applicability in real-time or resource-limited settings. Another limitation stems from its reliance on pre-trained text and image encoders, which could introduce biases and restrict the diversity of objects that can be generated [Jain et al., 2022]. The performance of Dreamfields is significantly influenced by the quality of these pre-trained models, which may vary. Furthermore, the model may struggle with textual descriptions that are inherently ambiguous or open to multiple interpretations. While designed to align closely with textual inputs, the current state of Dreamfields may find it challenging to handle overly vague or abstract descriptions effectively [Jain et al., 2022].

Dreamfields offer an innovative approach to 3D object generation through textual guidance. By leveraging a continuous volumetric representation and pre-trained encoders, Dreamfields achieve high-quality, detailed, and realistic 3D objects that closely align with the given textual descriptions. While the technology has its limitations, its advantages and potential applications make it a compelling avenue for future research and development in the field of automatic 3D model generation.

3.3.2 Dreamfusion

DreamFusion leverages Neural Radiance Fields (NeRFs) [Mildenhall et al., 2020] and uses a novel technique called Score Distillation Sampling (SDS) [Poole et al., 2022] which "generates high-fidelity coherent 3D objects and scenes for a diverse set of user-provided text prompts [Poole et al., 2022]. However, it is important to note that for the practical purposes of this thesis, *Stable* DreamFusion [Tang, 2022], is used, which is an open-source variant of DreamFusion. This variant introduces some differences compared to the original model. Stable DreamFusion modifies the original Imagen model by incorporating Stable Diffusion [Rombach et al., 2021], a latent diffusion model. It also employs the "multi-resolution grid encoder [from torch-ngp] to implement the NeRF backbone" [Tang, 2022] and uses the Adan optimizer as the default option [Tang, 2022].

DreamFusion, a product of Google Research, embodies a significant leap in the domain of text-to-3D synthesis, a realm where textual descriptions are converted into three-dimensional visual models. It builds upon recent advancements in text-to-image synthesis driven by diffusion models trained on extensive image-text datasets. Unlike its predecessors, DreamFusion adeptly navigates the challenges posed by the lack of large-scale datasets of labeled 3D assets and efficient architectures for denoising 3D data.

The genesis of DreamFusion lies in the evolution of Dream Fields, a generative 3D AI system unveiled by Google in late 2021. Dream Fields initially married OpenAI's image analysis model, CLIP, with Neural Radiance Fields (NeRF) to foster the generation of 3D views from text. DreamFusion further refined this approach by introducing a new loss based on Google's large AI image model, Imagen, thus paving the way for enhanced text to 3D synthesis.

At the core of DreamFusion's operation lies a fusion of Google's Imagen and NeRF's 3D capabilities. Imagen, a pre-trained 2D text-image diffusion model, forms the basis for text to 3D synthesis in DreamFusion. This synergy with NeRF, specialized for 3D generation tasks, enables the recovery and synthesis of new views of a particular scene from unobserved angles. DreamFusion employs a novel method termed Score Distillation

Sampling (SDS) to optimize a 3D scene given a text caption. SDS allows for the generation of samples from a diffusion model by optimizing a loss function. This method demonstrates the versatility of pre-trained image diffusion models as priors, enabling the optimization of samples in an arbitrary parameter space such as a 3D space, provided there is a mapping back mechanism. Understanding the underlying principles of Neural Radiance Fields (NeRF), Score Distillation Sampling (SDS), and the functionality of the Imagen model is crucial for a comprehensive grasp of DreamFusion's architecture. The primary application of DreamFusion is the generation of 3D models from textual descriptions. This capability has vast potential across various fields including, but not limited to, virtual reality, gaming, and educational domains where interactive 3D models can enhance user engagement and learning experiences.

Moreover, DreamFusion's ability to create relightable 3D objects and merge multiple 3D models into one scene opens avenues for more complex applications, enabling the creation of intricate, text-driven 3D scenes and animations.

Benefits: - **Data Efficiency**: DreamFusion circumvents the need for large-scale 3D labeled datasets, which are often a bottleneck in 3D synthesis projects. - **Generative Capability**: The generation of high-quality, relightable 3D objects based on textual input extends the boundaries of generative models.

Limitations: - **Model Maturity**: The generated 3D models, although promising, may not yet attain a high level of accuracy, indicating a scope for further refinement. - **Computational Resources**: The processing power required for the generation of 3D models could be substantial, posing challenges for resource-constrained environments.

DreamFusion marks a significant stride in bridging textual descriptions with 3D visualization using artificial intelligence. Its innovative architecture, built upon the synergy between Google's Imagen and NeRF, alongside the introduction of Score Distillation Sampling, lays a solid foundation for future advancements in text-to-3D synthesis. While the journey towards perfecting this technology continues, the potential applications and benefits of DreamFusion are vast and poised to have a lasting impact on the fields of computer vision and artificial intelligence.

Score Distillation Sampling (SDS) is a technique introduced in DreamFusion to generate samples from a diffusion model by optimizing a loss function, allowing for the optimization of samples in an arbitrary parameter space, such as a 3D space. The core idea is to leverage the structure of diffusion models to enable tractable sampling via optimization. This is achieved by optimizing over parameters θ such that $\mathbf{x} = g(\theta)$ appears as a sample from the frozen diffusion model. A differentiable loss function is employed where plausible images incur a low loss, and implausible images incur a high loss.

Mathematically, the process can be broken down into several steps:

1. **Loss Function Optimization**: - The objective is to minimize a diffusion training loss with respect to a generated datapoint $\mathbf{x} = g(\theta)$, expressed as:

$$\theta^* = \arg\min_\theta \mathcal{L}_{\text{Diff}}(\phi, \mathbf{x} = g(\theta))$$

- In this step, DreamFusion tries to find the best set of parameters (denoted by θ) that would generate a 3D model from text. It does this by minimizing a "loss function," which is a way to measure how far off the generated model is from what is desired. The goal is to adjust the parameters θ so that this loss is as small as possible.

2. **Gradient Computation**: - The gradient of $\mathcal{L}_{\text{Diff}}$ is given by:

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

- Here, $w(t)$ is a weighting term, $\hat{\epsilon}_\phi(\mathbf{z}_t; y, t)$ is the predicted noise, ϵ is the true noise, and $\frac{\partial \hat{\epsilon}_\phi(\mathbf{z}_t; y, t)}{\mathbf{z}_t}$ and $\frac{\partial \mathbf{x}}{\partial \theta}$ are Jacobian terms.

- To minimize the loss, DreamFusion needs to know in which direction to adjust the parameters θ. This is done by computing the gradient, which tells us the direction in which the loss function is increasing. By moving the parameters in the opposite direction, the loss can be decreased. In the formula, the terms involving $\hat{\epsilon}_\phi$ and ϵ are comparing the predicted noise to the actual noise in the model, which helps in understanding how to adjust the parameters to get a better model.

3. **Effective Gradient**: - To bypass the computation of certain Jacobian terms, an effective gradient is proposed:

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

- The original gradient computation can be quite complex and computationally expensive. So, an alternative, simplified version of the gradient is used to make the optimization process more manageable. This simplified gradient still gives a good direction to adjust the parameters θ to minimize the loss, but without some of the computational overhead of the original gradient computation.

In practice, the diffusion model predicts the update direction, obviating the need to backpropagate through the diffusion model. Hence, the model acts like an efficient, frozen critic that predicts image-space edits.

When compared to a method like Collaborative Score Distillation (CSD), it's noted that while SDS optimizes a single 3D representation to maintain a high likelihood as evaluated by the diffusion model, CSD tends to excel in capturing coherent geometry and allows for the learning of finer details. Moreover, CSD can produce diverse, high-quality samples without requiring changes in random seeds, indicating some of the areas where SDS might have room for improvement.

3.3.3 Point-E

Point-E operates through a two-stage generation process, resulting in point clouds. The initial stage involves the creation of an image using the text-to-image model GLIDE [Nichol et al., 2021], which was "fine tuned on 3D renderings" [Nichol et al., 2022]. In the next stage, known as the image-to-3D model, RGB point-clouds are created with a stack of diffusion models [Nichol et al., 2022].

3.3.4 Shap-E

Shap-E involves training an encoder which "produces a latent representation of a 3D asset" [Jun and Nichol, 2023]. In a second step, a diffusion prior on the obtained latent representations is trained, conditioning it on images or text descriptions to capture additional information [Jun and Nichol, 2023].

3.4 3D from Video

//TODO

3.4.1 NERfs

//TODO

Chapter 4

Comparative Study

4.0.1 Experimental Setup

details about data collection, preprocessing, and model training

4.0.2 Performance Metrics

discussion of evaluation criteria used in the comparative study

4.0.3 Results and Analysis

present findings and their interpretation

Chapter 5

Future Directions

//TODO

5.0.1 Emerging Trends in 3D Model Generation

explore current developments and new directions in the field.

5.0.2 Potential Research Directions

suggest areas for future research based on your study's findings.

Chapter 6

Conclusion

//TODO

6.0.1 Summary of Findings

provide a concise overview of the main results

6.0.2 Contributions to the Field

highlight the significance of the thesis

6.0.3 Implications and Practical Applications

discusses real-world impact of the findings.

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.

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.

Croitoru, F.-A., Hondru, V., Ionescu, R. T., and Shah, M. (2023). Diffusion models in vision: A survey.

Diggle, P. J. and Gratton, R. J. (1984). Monte carlo methods of inference for implicit statistical models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 46(2):193–212.

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., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale.

Gerats, B. G. A., Wolterink, J. M., and Broeders, I. A. M. J. (2023). Dynamic depth-supervised nerf for multi-view rgb-d operating room images.

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`.

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. *ArXiv*, abs/2006.11239.

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.

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

Jain, A., Mildenhall, B., Barron, J. T., Abbeel, P., and Poole, B. (2022). Zero-shot text-guided object generation with dream fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 867–876.

Jun, H. and Nichol, A. (2023). Shap-e: Generating conditional 3d implicit functions.

Kingma, D. P., Salimans, T., Poole, B., and Ho, J. (2023). Variational diffusion models.

Kingma, D. P. and Welling, M. (2022). Auto-encoding variational bayes.

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.

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.

Michelucci, U. (2022). An introduction to autoencoders.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2020). Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*.

Nichol, A., Dhariwal, P., Ramesh, A., Shyam, P., Mishkin, P., McGrew, B., Sutskever, I., and Chen, M. (2021). Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*.

Nichol, A., Jun, H., Dhariwal, P., Mishkin, P., and Chen, M. (2022). Point-e: A system for generating 3d point clouds from complex prompts. *arXiv preprint arXiv:2212.08751*.

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

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

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. (2021). Learning transferable visual models from natural language supervision.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models.

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. (2021). High-resolution image synthesis with latent diffusion models.

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.

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., 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, L., and Polosukhin, I. (2023). Attention is all you need.

Wei, Y., Liu, S., Rao, Y., Zhao, W., Lu, J., and Zhou, J. (2021). Nerfingmvs: Guided optimization of neural radiance fields for indoor multi-view stereo.

Xiao, Z., Kreis, K., and Vahdat, A. (2022). Tackling the generative learning trilemma with denoising diffusion gans.

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*.

Yariv, L., Kasten, Y., Moran, D., Galun, M., Atzmon, M., Basri, R., and Lipman, Y. (2020). Multiview neural surface reconstruction by disentangling geometry and appearance.

Erklärung

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

Datum	Unterschrift