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 work 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 makes a significant contribution 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. Various forms of 3D data representation are explored while essential generative models such as Variational Autoencoders (VAEs), Generative Adversarial Networks (GANs), Diffusion Models, Contrastive Language-Image Pre-training (CLIP), and Neural Radiance Fields (NeRFs) are explained. These fundamental concepts provide the necessary foundation for a subsequent in-depth analysis of 3D model generation techniques.

The investigation also extends to the evaluation of different approaches to 3D model generation. 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 investigation 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.

Chapter 2

Basics

Before delving into the comparative study of automatic 3D model generation techniques, it is crucial to establish a common ground of understanding for the underlying technologies that power these generative methods. This section aims to provide an introductory overview of key concepts and models in the realm of generative machine learning and 3D data representation. The section starts with Variational Autoencoders (VAEs), which serve as probabilistic frameworks for learning complex data representations. Generative Adversarial Networks (GANs) are then discussed, highlighting their unique training dynamics that include both generator and discriminator components. The section further explores the domain of Diffusion Models, focusing on Denoising Diffusion Probabilistic Models (DDPMs), Stochastic Gradient Methods (SGMs), and Stochastic Differential Equations (SDEs), each with their unique approach to data generation. Additionally, the innovative field of Contrastive Language-Image Pre-training (CLIP) is examined, showcasing its ability to bridge natural language and visual data. This provides promising avenues for 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.

2.1 Variational Autoencoders - VAEs

Based on the seminal work by Diggle and Gratton, generative models can be classified into two broad categories. Prescribed models employ a well-defined, often parametric, mathematical expression for the probability density function (pdf), which enables easier analytical interpretation of the distributions. In contrast, implicit models synthesize new data samples without relying on an explicit pdf, approximating the underlying data distribution on which they were trained [Diggle and Gratton, 1984]. Variational Autoencoders (VAEs), which inherit the foundational architecture of autoencoders, belong to the prescribed models category as they require an explicit formulation of the probability density function (pdf) to function effectively. 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]. Generative Adversarial Networks (GANs) [Goodfellow et al., 2020], discussed later, are a prime example ofthe latter category.

VAEs inherit the fundamental architecture of autoencoders, consisting of an encoder and a decoder. The encoder aims to transform the input data into a lower-dimensional latent space representation (vector), often referred to as the "code" or "bottleneck" [Hinton and Salakhutdinov, 2006; Goodfellow et al., 2016]. This code captures the most relevant features of the input while reducing its dimensionality. The decoder then reconstructs the original input data from the given latent vector using a loss function. However, the core objective of an autoencoder is not the reconstruction itself but the extraction of a meaningful latent vector [Goodfellow et al., 2016], which serves as a simplified representation of the input."Autoencoders have been successfully applied to dimensionality reduction and information retrieval tasks." [Goodfellow et al., 2016]. The ability to reduce dimensionality has practical implications for improving the efficiency of classification tasks by reducing computational and memory overhead [Goodfellow et al., 2016]. When paired with information retrieval, this dimensionality reduction makes searching in certain low-dimensional spaces particularly efficient [Goodfellow et al., 2016]. Despite these advantages, autoencoders are not designed to generate new data; their main function is to copy and reconstruct the given input [Goodfellow et al., 2016].

while similiar inputs tend to cluster in similiar latent spaces. It can be seen that the same digits tend to cluster themselves in the latent space. Another important thing to note is that there are parts of the latent space that doesn't correspond to any data point. Using those as inputs to the encoder will result in an output that doesn't look like any digit from the MNIST data. This is what we mean by that the latent space is not regularized. Such a latent space only has a few regions/cluster that has the generative capability, which means that sampling any point in the latent space that belongs within a cluster will generate a variation of the data that the cluster belongs to. But the entire latent space does not have the generative capability. The regions which do not belong to any cluster will generate garbage output. Once the network is trained, and the training data is removed, we have no way of knowing if the output generated by the decoder from a randomly sampled latent vector is valid or not. Hence AE is mainly used for compression - https://towardsdatascience.com/difference-between-autoencoder-ae-and-variational-autoencoder-vae-ed7be1c038f2

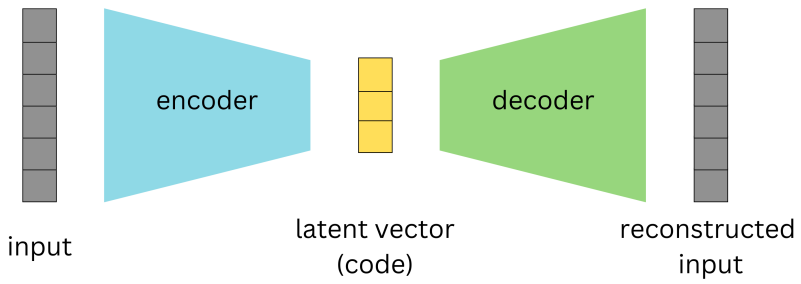

Figure 1: Autoencoder: The encoder reduces the input dimension to a latent vector that captures the most important features. The decoder then uses this vector to reconstruct the input, with training aimed at minimizing reconstruction loss.

In VAEs, the encoder compresses the input into a latent variable space and creates a probability distribution over the latent variables. This allows for robustness against overfitting and enables the model to synthesize new, analogous data points by sampling from this distribution and subsequent decoding [Kingma and Welling, 2022; Rezende et al., 2014].During the training phase, VAEs identify specific regions in the latent variable space, akin to "pools," for different categories of data, such as various types of animals. These regions enable VAEs to overcome a limitation inherent in standard autoencoders—the uncertainty of where to sample a useful latent vector for the decoder. In VAEs, this uncertainty is mitigated by constraining the latent space to known regions from which vectors can be confidently sampled [Doersch, 2016]. The decoder in VAEs reconstructs the original input or synthesizes new outputs from sampled latent variables, optimized via a loss function that considers both reconstruction loss and a regularization term based on the Kullback-Leibler (KL) divergence [Goodfellow et al., 2016]. This divergence measures differences between the estimated and true data distributions [Kingma and Welling, 2022], aiding the model in generalizing effectively to unseen data

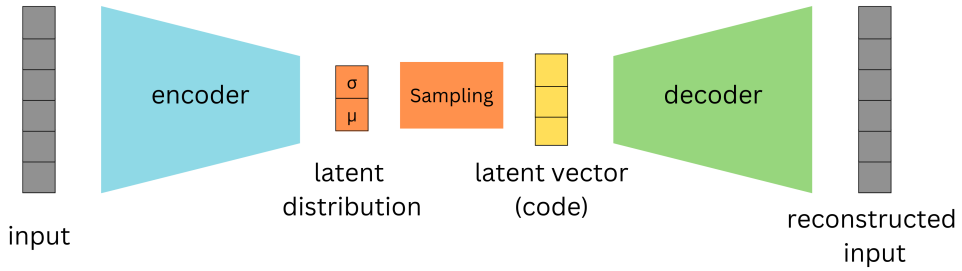

Figure 2: Functionality of a Variational Autoencoder, demonstrating incorporation of mean and standard deviation for enhancing generative capabilities.

Despite their capabilities, VAEs exhibit some limitations. According to Goodfellow et al., the generated samples can often be blurry. He explains that 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 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 [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 hyper-parameter 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. Instead, this approach focuses on finding inherent relationships, similarities, or clusters within the data itself. 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.

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

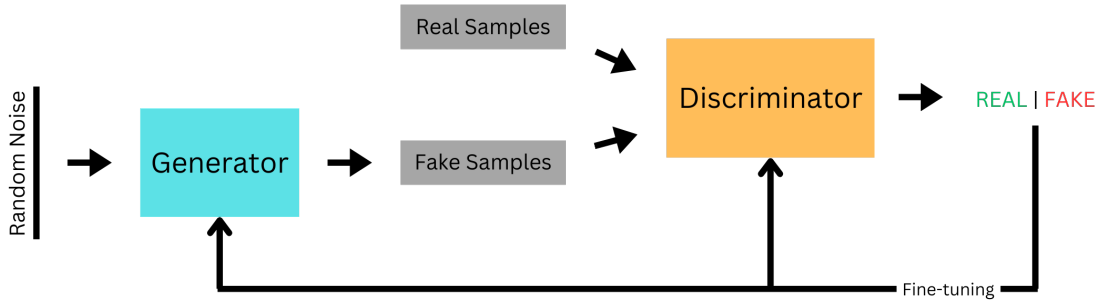

Figure 3: Simplified functionality of a Generative Adversarial Network

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]. Additionally, Brophy et al., highlight three significant issues commonly associated with GANs. 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 [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.

2.3 Diffusion models

The limitations of GANs, which were just stated above, have led to the emergence of diffusion models, a class of AI tools that offer distinct advantages over traditional generative models. These models differ from GANs in their approach to modeling and generating data. Instead of explicitly modeling the joint distribution of data and noise variables, diffusion models operate by progressively perturbing data with noise and then learning to reverse this process to generate new samples.

In current research, there are three main approaches that dominate the study of diffusion models: Denoising Diffusion Probabilistic Models (DDPMs) [Ho et al., 2020], 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 [Sohl-Dickstein et al., 2015]. The forward chain introduces noise to the data using a normal distribution. This iterative process continues until the data converges to a pure isotropic Gaussian 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].

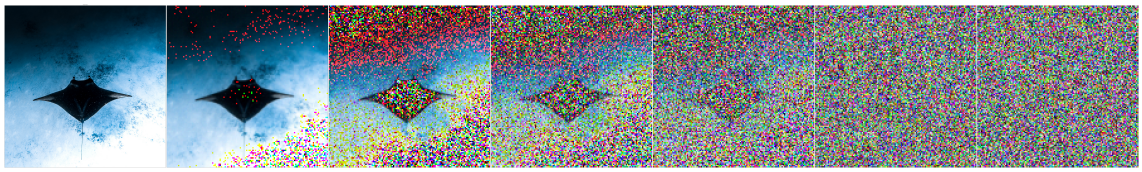

Figure 4: Forward process adding noise to an image

The reverse chain, in turn, uses an adaptive parameterized deep neural network to progressively remove the noise added by the forward chain. This iterative process aims to generate data patterns that closely resemble the original data by gradually reducing the noise. Training the reverse chain involves minimizing the Kullback-Leibler (KL) divergence between the joint distributions of the forward and reverse chains and maximizing the variational lower bound (VLB) of the log-likelihood of the data. The KL divergence measures the dissimilarity between two probability distributions, in this case, the forward and reverse chains. By minimizing the KL divergence, the reverse chain is trained to approximate the inverse of the forward process, effectively removing the noise added by the forward chain [Sohl-Dickstein et al., 2015]. The VLB provides a lower bound approximation of the log-likelihood, a measure of how well the model captures the underlying data distribution. Maximizing the VLB ensures that the reverse chain generates data patterns that closely match the original data distribution. [Ho et al., 2020; Sohl-Dickstein et al., 2015]. It's important to clarify that in the context of diffusion models, VLB is a term associated with variational inference, and its usage might not be directly equivalent to VAEs. Regarding what the reverse network predicts, it typically estimates the parameters of a conditional distribution that represents how to transform the noisy data at each step to recover the original data. This prediction guides the process of removing the noise introduced during the forward chain.

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. 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.

2.3.2 Score-Based Generative Models

SGMs are a class of diffusion models that have gained prominence in the field of generative modeling due to their ability to capture complex data distributions and generate diverse and realistic samples.

Score-based generative models (SGMs) 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 score 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]. To put it simply, the score can be seen as a vector field, indicating the direction in which the logarithm of the data density experiences the most significant growth [Song and Ermon, 2019]. In essence, the score function reveals how the data distribution responds to small variations within the data itself, serving as a guiding principle for the generative model. To train a neural network for SGMs, Song and Ermon [2019] employ a technique called score matching [Hyvärinen and Dayan, 2005]. This involves estimating the score function for data points intentionally perturbed with Gaussian noise. This means that score matching effectively learns how to denoise the noisy data and restore the original data distribution. [Song and Ermon, 2020].

In order to generate Samples, Langevin Dynamics [Roberts and Tweedie, 1996] is used, which simulates a particle's movement in a field of potential energy, where the score function serves as the guiding force. By moving data samples along the direction of the score function, Langevin Dynamics effectively 'pulls' them toward areas where data density is higher, essentially places where they blend better with the overall dataset.

However, 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 ensures that the resulting data distribution doesn't condense into a lower-dimensional structure [Song and Ermon, 2019].

2.3.3 Stochastic Differential Equations

Score SDEs provide a flexible framework for modeling and generating complex data distributions with inherent stochasticity. Unlike explicit modeling of the joint distribution of data and noise variables, SDEs focus on describing the dynamics of a diffusion process through differential equations. These equations incorporate both deterministic components that govern the overall trend of the process and stochastic components that account for the inherent randomness in the data generation process. In Score SDEs, the diffusion

process is represented as the continuous-time evolution of a random variable. By specifying the drift and diffusion coefficients within the SDEs, one can capture the behavior and evolution of the data distribution over time. The stochastic nature of SDEs allows for modeling intricate dependencies and capturing complex patterns in the data. Training SDEs involves estimating the parameters of the drift and diffusion coefficients. This is typically done by maximizing the likelihood of the observed data under the SDE framework. Various estimation techniques, such as maximum likelihood estimation or Bayesian inference, can be employed to optimize the parameters. SDEs can capture long-term dependencies and accurately model the evolution of data over time. Additionally, the stochastic nature of SDEs enables them to generate diverse and realistic samples by incorporating inherent randomness into the data generation process.

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

NeRF - Acronym for Neural Radiance Fields. Neural as there a neural network. Radiance as the neural network describes the radiance of the scenes (how much light is been emitted by a point in space and in each direction). Field as this is a continuous function - a smooth thing that exists in the world.

Neural Radiance Fields (NeRFs) [Mildenhall et al., 2020] offer an alternative approach to representing 3D scenes from a limited set of 2D images, as introduced by Mildenhall et al.. Unlike traditional 3D reconstruction methods, NeRFs use a volumetric representation to accurately capture both the spatial and angular distribution of light. Traditional techniques often fall short in capturing complex geometries and variable lighting conditions, issues that NeRFs successfully address by modeling the volumetric scene as a continuous 5D function. This function accepts 3D spatial coordinates and viewing direction as inputs and produces the color and opacity at a specific point in the scene. This approach contrasts with classic deep learning methods, which require a comprehensive dataset comprising various scenes and their representations. NeRFs, however, are trained to specialize in a single, unique scene [Mildenhall et al., 2020]. The neural network consists of layers specifically designed to encode the volumetric details of that particular scene, effectively creating a dedicated neural network for each scene.

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

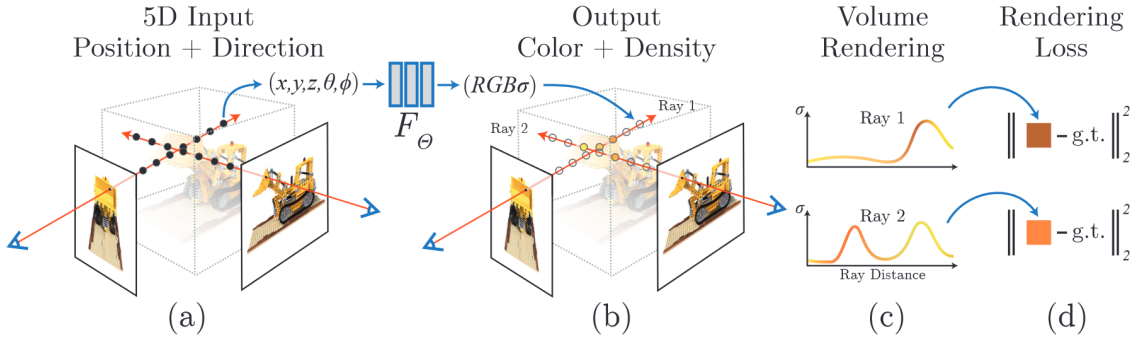

Figure 5: Neuronal Radiance Field. Figure taken From Mildenhall Mildenhall et al. [2020] - Figure 2

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 d. 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 enables capturing complex visual phenomena like lighting, reflections, and transparency. For each pixel in a desired image frame, a similar querying process is performed. 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 5. The point at which this density curve rises ignificantly 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. 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. The idea is that density, or transparency, is invariant to the direction from which an object is viewed. 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 and SLAM can address this issue. Once these are identified, new views can be synthesized by querying the neural network for color and density information along rays projected through the scene. 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 alone these rays, and integrating the sampled values to compute a pixel's color, is differentiable.

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

$$C(r) = \int_{t_n}^{t_f} T(t)\sigma(r(t))c(r(t), d)dt$$

, where

$$T(t) = \exp\left(-\int_{t_n}^{t} \sigma(r(s))ds\right)$$

This formula allows for the rendering of a 3D scene by casting rays and integrating their properties from a near boundary to a far boundary. Specifically, $T(t)$ represents the probability of a ray passing through empty space up to a certain point, influenced by the density function σ. This calculation determines whether the ray continues through empty space or ends when it hits an object, with color adjustments made at this endpoint. The model can also capture complex details such as transparency. Although Naive NeRF models cannot provide photorealistic results due to the lack of detail, several optimizations have been introduced to improve their performance. One of these improvements is the use of position encoding techniques that deterministically map 3D coordinates and view directions to a higher dimensional space using a hierarchical series of sine and cosine functions. This approach differs from position coding in transformer models and is used to capture both fine and coarse detail in 3D space. Another optimization involves hierarchical volume sampling by a two-level neural network system consisting of "coarse" and "fine" layers. The coarse network initially sparsely samples the rays in the 3D scene and provides instructions to the fine network for more detailed sampling. While both meshes are optimized simultaneously, the final rendering is based solely on the fine mesh.

The experimental results indicate several key strengths and limitations of the Neural Radiance Fields (NeRF) approach. One of its notable advantages is the model's ability to handle fine-grained structures, which is particularly evident when examining the details of rendered objects like a microphone. Another significant benefit is the model's efficiency in terms of memory usage. For example, a single scene rendered using NeRF only occupies about five megabytes of memory, a stark contrast to voxel grid representations that can consume over 15 gigabytes for the same scene. Intriguingly, the memory footprint of the rendered scene is even smaller than that of the input images, making the model highly efficient for data storage and transmission. However, the model is not without drawbacks. One limitation is the computational cost involved in training the neural network. The optimization process for a single scene can require around 100,000 to 300,000 iterations to converge when using a single NVIDIA V100 GPU, translating to a training time of approximately one to two days. Although this doesn't necessitate a data center, it does require a significant amount of time. The authors also conducted ablation studies to understand the contributions of various model components. The results underscore the importance of positional encodings and view-dependent effects. Omitting positional encodings or view dependence negatively impacts the model's ability to capture realistic light effects and fine-grained structures, respectively. Therefore, these components are crucial for the model's performance and its ability to render scenes with high fidelity.

- Limited rendering model: difficult to optimize + Highly compressible (1-10MB)

3.2 3D from Image

//TODO

3.3 3D from Text input

//TODO

3.3.1 Dreamfields

The concept proposed by Jain et al. was groundbreaking, as they introduced the innovative approach of training a 2D diffusion model on images to learn 3D structure. This approach eliminated the reliance on pre-existing 3D Models, which is often scarce in large datasets.

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].

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

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.

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.

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.

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. 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.

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*.

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. and Ermon, S. (2020). Improved techniques for training score-based generative models.

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.

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*.

Erklärung

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

Datum	Unterschrift