UKRAINIAN CATHOLIC UNIVERSITY

MASTER THESIS

Text-Guided 3D Synthesis with Latent Diffusion Models

Author:
Danylo KOVALENKO

Supervisor:
Oles PETRIV

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2023

Declaration of Authorship

I, Danylo KOVALENKO, declare that this thesis titled, "Text-Guided 3D Synthesis with Latent Diffusion Models" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Master of Science

Text-Guided 3D Synthesis with Latent Diffusion Models

by Danylo KOVALENKO

Abstract

The emergence of diffusion models has greatly impacted the field of deep generative models, establishing them as a powerful family of models with unparalleled performance in various applications such as text-to-image, image-to-image, and text-to-audio tasks. In this work, we aim to propose a solution for text-guided 3D synthesis using denoising diffusion probabilistic models, while minimizing the memory and computational requirements. Our goal is to achieve high-quality and high-fidelity 3D object generation conditioned by text or a label in a number of seconds. We propose to use a triplane space parametrization in combination with a Latent Diffusion Model (LDM) to generate smooth and coherent geometry. The LDM is trained on the large-scale text-3d dataset and is used as a latent triplane texture generator. By using a triplane space parametrization, we aim to improve the efficiency of the space representation and reduce the computational cost of synthesis. We will also give a theoretical justification that this kind of parametrization of 3d space is capable of containing not only information about the geometry but also about the color and reflectivity of the figure. Additionally, we use an implicit neural renderer to decode geometry details from triplane textures.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The use of diffusion models [Ho, Jain, and Abbeel, 2020; Nichol and Dhariwal, 2021; Song, Meng, and Ermon, 2020] is a growing trend in the generation of high-quality and diverse content, such as images [Ramesh et al., 2022; Saharia et al., 2022; Rombach et al., 2021], audio [Schneider, Jin, and Scholkopf, 2023], and video[Ho et al., 2022]. Diffusion models have several benefits over other generative models, including better coverage of data distributions, and more stability during training, research shows that they beat many state-of-the-art models of those times [Dhariwal and Nichol, 2021]. Most diffusion models are used to generate 2D images by working with their representation as pixel grids [Saharia et al., 2022] or as compressed latent space [Rombach et al., 2021; Ramesh et al., 2022] that are constructed by the autoencoders. However, recently researchers have started to apply diffusion generative models to synthesize 3D content [Poole et al., 2022; Lin et al., 2022; Gao et al., 2022; Zeng et al., 2022], which has a less standardized representation, including voxels, point clouds, meshes, and implicit functions. This variety of representations makes the application of diffusion models to 3D shapes less straightforward than in 2D, however, there are a number of works showing the pros and cons of each.

The research community has proposed a wide range of solutions to address the problem of 3D synthesis. These solutions can broadly be grouped into three main categories:

Diffusion models as prior: Solutions [Poole et al., 2022; Lin et al., 2022] that utilize generative diffusion models as a prior model for various neural implicit scene representations, such as neural radiance fields (NeRFs) [Mildenhall et al., 2020]. In this method, the diffusion model is pre-trained on the text-to-image pairs and used as guidance for the NeRF model by propagating gradients into NeRF weights. Generating one scene with a text prompt usually takes quite a lot of time. Due to the temporal inconsistency of the diffusion signal to the camera position and due to the lack of information about 3D scenes, as well as due to the long convergence time of nerf models even in a fairly stable configuration, the quality of the resulting scene is rather low.

Compressed 3D Representations: Approaches that focus on studying the representation of 3D space in a more compressed format, such as triplanes [Shue et al., 2022; Wang et al., 2022; Gao et al., 2022; Gupta and Gupta, 2023; Wu et al., 2023] or signed distance functions (SDFs) [Nam et al., 2022]. As a rule, in such approaches, the authors suggest using an explicit-implicit representation of a 3D scene [Müller et al., 2022; Chen et al., 2022a]. Where meta-information about the scene is stored as a latent texture or parameters, which is subsequently decoded by some kind of implicit neural network to achieve 3D geometry. After all, the diffusion models learn how to generate these textures, to synthesize novel figures.

3D Diffusion models: The final category includes works that propose the direct synthesis of a 3D scene using a 3D diffusion process [Zeng et al., 2022; Nichol et al., 2022; Chou, Bahat, and Heide, 2022; Zhang et al., 2023]. These approaches often suggest the use of point clouds as a representation of the 3D scene. A randomly initialized point cloud is iteratively optimized to reach a state that describes geometry. As a rule, after that, there is still a lot of post-processing work to convert the point cloud into a mesh. The main problem with these approaches is the complexity of their convergence and parameterization. Also, such approaches take up a lot of memory due to multi-stage generation, which uses a large number of parameters.

Research of current methods in the field of 3D synthesis has shown that most approaches are either conditioned very poorly or generate 3D scenes in an unconditioned manner due to the lack of large, representative datasets in the form of text-3D pairs.

In this work, we will focus more on the direction in using explicit-implicit [Müller et al., 2022; Chen et al., 2022a] models but in a point cloud setting, that is parametrized by a diffusion generative model that operates in a latent space of variational auto-encoder. We were inspired by the idea to improve solutions by taking only the best ideas from such works as [Shue et al., 2022; Zheng et al., 2022; Chou, Bahat, and Heide, 2022; Nam et al., 2022; Wang et al., 2022; Gupta and Gupta, 2023]. Our choice is motivated by the idea, that such a pipeline could be easily broken into two major parts that can be sequentially but independently optimized. Also, such implicit architecture requires a low memory footprint to learn how to reconstruct high-frequency features from a compressed state. We believe, that our model will have a small number of parameters to be capable make inference on small GPUs in a number of seconds. An open question in our research is about the most compressed and efficient 3D space explicit-implicit representation that will be compatible with 2D latent diffusion models.

We suggest employing 2D triplane textures, which are a type of hidden representation that can capture information about the fine details of a 3D shape. These textures have demonstrated the ability to compress 3D information effectively and with high quality. Following this, triplanes are decoded by a lightweight MLP (Multilayer Perceptron) model to reconstruct the 3D object.

In this work, we introduce the concept of adapting LDM(latent diffusion model), a generative model that can handle conditioning through text, class labels, or even images to generate triplane textures. We will highlight the effectiveness of this model by using label conditioning. Moreover, with sufficient training resources, there is the potential to develop the model to a point where it can be exclusively guided through text.

Our goal is to implement 2 main features in the pipeline, which are iterated according to the complexity of implementation: (i) we want to be able to set the condition in the form of text or label to generate geometry by the description from the user, which will make it possible for artists to use this solution in the real world tasks, (ii) provide the ability to convert 2D images in the 3D mesh.

In section [2] we will briefly discover existing solutions for a combination of 2D diffusion models and 3D neural fields. In section [3] we will discover proposed improvements to current explicit-implicit (see section [3.1]) 3D synthesis by applying prior from state-of-the-art latent diffusion models (see section [3.2]) and training strategy (see section [5.1.2]) that we want to apply. After all, we will discuss potential challenges and summarize our research plan in section [6].

To summarize our contribution is:

1. We introduce an efficient solution for compressing 3D scenes by employing an explicit form of parameterization, which is then followed by an implicit decoding process that reconstructs the scenes by point clouds.
2. Propose to reuse the latest state-of-the-art diffusion model architecture in a class or text conditional manner that will be capable to synthesize novel explicit parameterizations for implicit decoder model.
3. Present a multi-stage training pipeline strategy on large and diverse 3D datasets.

Chapter 2

Related Work

This work combines state-of-the-art latent diffusion models with implicit 3D neural fields to achieve high-quality and conditional 3D scene representation. We provide an overview of the related work in the areas of diffusion models and leading-edge 3D scene representation.

2.1 Diffusion models.

Generative models, such as Generative Adversarial Networks (GANs) [Goodfellow et al., 2014; Mirza and Osindero, 2014; Karras et al., 2019] and autoregressive models [Razavi, Oord, and Vinyals, 2019], have been well-established in the field of generative models. Latent space models have shown significant improvements in synthesis speed, especially when pretraining the VAE with a Gaussian prior. Nevertheless, Denoising Diffusion Probabilistic Models (DDPMs) [Ho, Jain, and Abbeel, 2020; Nichol and Dhariwal, 2021] and their latent space modifications [Rombach et al., 2021] currently hold the state-of-the-art in 2D image synthesis due to their ability to generate high-quality and diverse images while offering stable training and capturing the full training distribution. GANs, on the other hand, can be challenging to train and may experience mode collapse [Dhariwal and Nichol, 2021].

2.2 Diffusion Models in 3D Generation

Several methods have been proposed to utilize 2D diffusion models as priors for 3D synthesis. One such approach [Nam et al., 2022] suggests using DDPM [Ho, Jain, and Abbeel, 2020] to generate the parameters of the signed distance function (SDF) in a conditional manner with CLIP (Contrastive Language-Image Pre-Training) [Radford et al., 2021] embeddings. This approach requires fewer trainable parameters, allowing the models to run on smaller video cards, but lacks explicit representation of 3D space, leading to artifacts in the final result.

Recent advances in text-to-image synthesis have been driven by diffusion models trained on large-scale image-text datasets [Schuhmann et al., 2022]. However, applying this approach to 3D synthesis is challenging due to the lack of large-scale labeled 3D datasets and efficient architectures for denoising 3D data. In "DreamFusion: Text-to-3D using 2D Diffusion"[Poole et al., 2022], the authors propose a novel approach to overcome these limitations by using a pre-trained 2D text-to-image diffusion model as a prior for optimizing a parametric 3D model, Neural Radiance Field (NeRF) [Mildenhall et al., 2020], using a probability density distillation-based loss function. This method demonstrates the effectiveness of pre-trained image diffusion models as priors, allowing the generation of 3D models of the given text that can be viewed from any angle and composited into any 3D environment.

In another work, "Magic3D" [Lin et al., 2022], the authors address the limitations of DreamFusion [Poole et al., 2022] by leveraging a two-stage optimization framework. Magic3D synthesizes 3D content with 8x higher resolution supervision and is 2x faster than DreamFusion [Poole et al., 2022]. However, both DreamFusion and Magic3D still require multiple GPU-hours of inference time for one sample, which is considerably longer than state-of-the-art generative image models.

The authors of "Point·E: A System for Generating 3D Point Clouds from Complex Prompts" [Nichol et al., 2022] propose an alternative method for 3D object generation, which produces 3D models in only 1-2 minutes on a single GPU by combining recent methods for text-to-3D synthesis. However, while this model is a meaningful step towards fast text-to-3D synthesis, it has limitations in producing colored three-dimensional shapes at a relatively low resolution in a 3D format (point clouds) that does not capture fine-grained shape or texture. Extending this method to produce high-quality 3D representations such as meshes or NeRFs could allow the model's outputs to be used for a variety of applications. However, point clouds have limitations such as poor surface continuity, which is the main disadvantage of this model.

In the paper "3D Neural Field Generation using Triplane Diffusion" [Shue et al., 2022], the authors propose preprocessing training data, such as ShapeNet meshes, by converting them to continuous occupancy fields and factoring them into a set of axis-aligned triplane feature representations. This approach allows them to train existing 2D diffusion models, specifically Denoising Diffusion Probabilistic Models (DDPM) [Ho, Jain, and Abbeel, 2020] on these representations to generate 3D neural fields with high quality and diversity, outperforming alternative approaches to 3D-aware generation. In our work, we aim to propose a novel approach for 3D generation by directly generating triplanes parameterization of implicit SDF (signed distance function) with more recent state-of-the-art 2D latent diffusion models [Rombach et al., 2021] with pre-train on large and diverse dataset like Objaverse. This grants the model near-complete control over the generated neural field and allows us to treat well-fit triplanes in a shared latent space as ground truth data. We also want to answer the question of whether this grounded spatial latent space can make it real to learn diverse data objects in compressed latent form, followed by implicit decoding to point clouds.

In recent times, a novel algorithm named "3DGen: Triplane Latent Diffusion for Textured Mesh Generation" was presented [Gupta and Gupta, 2023]. The proponents of this work leveraged a 3D Variational Autoencoder (VAE), a model proficient in compacting 3D space into a less intricate dimension—triplane textures—and deciphering them back into a colored mesh. They emphasized the criticality of employing unique 3D convolutions for the generation of triplane textures, as advocated in the related work [Wang et al., 2022]. Moreover, they introduced the use of a differentiable marching tetrahedra algorithm, as detailed in another study [Gao et al., 2022], to facilitate the creation of high-quality meshes. Despite its recent publication, the authors suggested training the model on expansive datasets, such as "Objaverse" [Deitke et al., 2022]. They put forth the use of a latent diffusion model for the generative component, recognizing it as the current pinnacle in content generation. This work, although it was unveiled substantially after the commencement of our research, has provided us with profound insights. These include the essential role of triplane regularizations, the intricacies involved in training latent diffusion models, and the application of VAE in learning low-dimensional latent representations of high-dimensional spaces such as 3D. In our work, we want to propose a similar pipeline, however, simplifying the process of final decoding of 3D figures, as suggested in the work "3D Neural Field Generation using Triplane Diffusion" [Shue

et al., 2022], as well as the generation of triplines.

Finally the latest work from NVIDIA "LION: Latent Point Diffusion Models for 3D Shape Generation"[Zeng et al., 2022] proposes to operate only in 3D point clouds. The authors introduce the hierarchical Latent Point Diffusion Model (LION) for 3D shape generation. LION is set up as a variational autoencoder (VAE) with a hierarchical latent space that combines a global shape latent representation with a point-structured latent space. For a generation, they train two hierarchical denoising diffusion models (DDMs) in these latent spaces. The hierarchical VAE approach boosts performance compared to DDMs that operate on point clouds directly, while the point-structured latents are still ideally suited for DDM-based modeling.

2.3 Leading-edge 3D Scene Representation.

Neural fields [Mildenhall et al., 2020; Müller et al., 2022; Chen et al., 2022a; Karnewar et al., 2022; Barron et al., 2021; Suhail et al., 2021; Sitzmann et al., 2021], or implicit neural representations, are at the forefront of 3D scene representation. These fields can learn both geometry and appearance through posed images or just geometry alone. Neural fields represent scenes as continuous functions, making them more scalable and flexible for complex scenes than their discrete counterparts. In the past, a single large MLP [Mildenhall et al., 2020] was utilized to represent entire scenes, but this approach had limited efficiency as it required many forward passes through the model during training. Recent advancements [Müller et al., 2022; Chen et al., 2022a; Suhail et al., 2021; Sitzmann et al., 2021] have focused on locally conditioned representations that use small MLPs, which are efficient during inference and better at capturing local details in the scene. Our work adopts the hybrid triplane-vector representation that has proven [Chen et al., 2022a; Müller et al., 2022] to be expressive and efficient, scaling with surface area and integrating well with 2D generator architectures. Our modifications enhance its compatibility with denoising techniques.

In conclusion, this work combines state-of-the-art latent diffusion models [Rombach et al., 2021] with implicit 3D neural fields [Chen et al., 2022a] modifications in point cloud settings to achieve high-quality and conditional 3D scene representation.

Chapter 3

The Proposed Method

Previous research work "3D Neural Field Generation using Triplane Diffusion" [Shue et al., 2022] utilizes occupancy fields as a representation of space. Occupancy fields are randomly initialized point cloud that is then processed by a neural network to differentiate between geometry and non-geometry points. The authors of this work propose to parameterize the implicit neural network using triplane parameterization, which yields fast convergence but also poses a significant memory requirement challenge. They uniformly generate ten million points in a space, however, only a small portion of these points were used in the final geometry result. Additionally, this work used an outdated diffusion model [Ho, Jain, and Abbeel, 2020; Nichol and Dhariwal, 2021](DDPM), while there are more robust and effective models available, such as the latent diffusion model [Rombach et al., 2021].

In our current study, we aim to enhance the previous approach by incorporating a more advanced state-of-the-art latent diffusion model [Rombach et al., 2021] instead of the conventional denoising diffusion probabilistic model (DDPM) [Ho, Jain, and Abbeel, 2020]. We also intend to improve the triplane parametrization by factorizing space into three planes and three vectors, as suggested by the authors of [Chen et al., 2022a]. Most notably, we propose to minimize memory consumption by replacing the occupancy field with a signed distance function (SDF) defined as $sdf : \mathbb{R}^3 \rightarrow \mathbb{R}$ that maps a 3D coordinate to its shortest distance to the nearest surface, is employed to imbue the triplane texture with geometric complexity. This function will iteratively move the randomly initialized point cloud toward the geometry, thereby preserving information and allowing for explicit control over the number of 3d points to be optimized.

The problem we aim to address in this paper can be broken down into the following two main components:

1. Compressed 3D Representation: One of the major challenges in the field of 3D space representation is to find a compact and efficient way to represent complex and vast 3D space. In this work, we aim to propose a solution that can effectively capture the intricacies of 3D space while minimizing the memory and computational requirements.

2. Generative Training Approach: The state-of-the-art latent diffusion model (LDM) [Rombach et al., 2021] has been shown to be a powerful tool for 3D scene generation. However, to fully realize its potential, we aim to adapt the LDM to effectively generate 3D scenes from class labels. This requires an in-depth understanding of the model and a thorough evaluation of its suitability for this particular task.

3.1 Compressed 3D Representation

In this work, we specifically focus on 3D scene representations using parametrized implicit signed distance function model. The output of the implicit SDF, in this case, represents a distance indicating how far away a point from the object (see Figure 3.1 for more model architecture details).

Intuition. Triplane space parametrization draws on the principle of projecting sampled points from a 3D object onto three orthogonal planes, thereby creating a compact yet efficient representation of the object's geometry. This hybrid approach combines elements from both explicit and implicit representations, inheriting their strengths while mitigating their weaknesses.

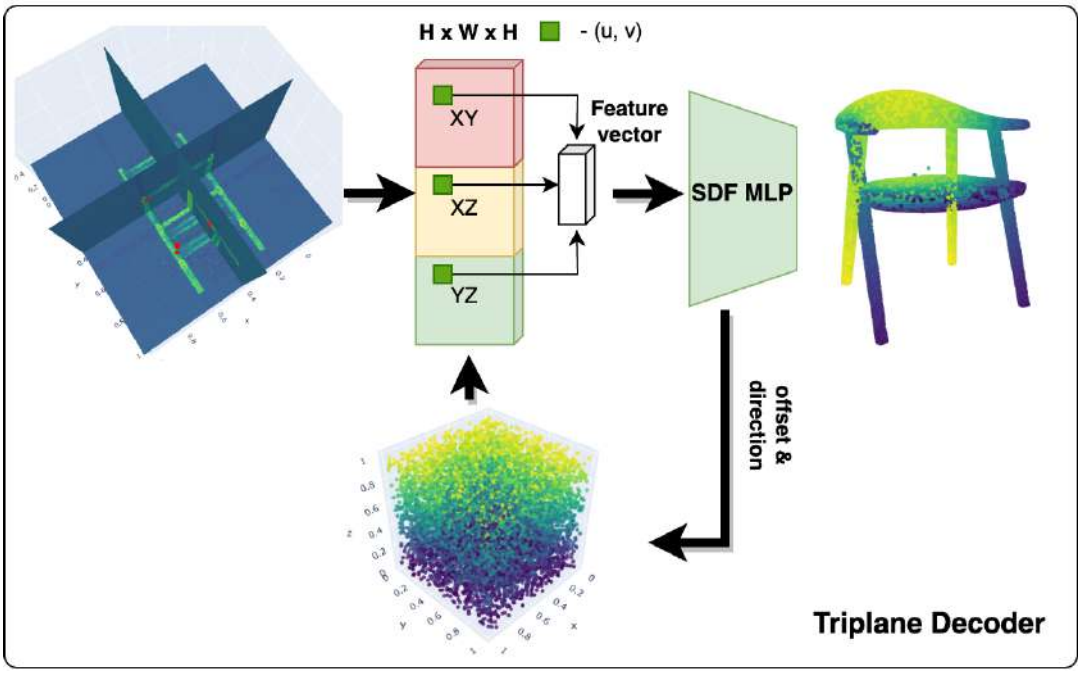

FIGURE 3.1: This illustration shows the full pipeline of projecting a point onto three orthogonal planes, sampling features and their aggregation, and then decoding the resulting features into an SDF values using a lightweight neural network.

Explicit 3D representations, such as voxel grids, are computationally fast but memory-intensive due to the necessity of storing the entire voxel grid. Implicit representations, like NeRF, offer memory efficiency by representing the scene as a continuous function, but at the cost of expensive predictions. The triplane representation bridges these techniques by storing just three planes in memory and projecting the 3D positions onto these planes, achieving both prediction speed and memory efficiency.

Forming the Triplane Textures. As a parametrization of the signed distance function model, we decided to use a triplane-vector parametrization, as proposed in [Chen et al., 2022a, TensoRF: Tensorial Radiance Fields], which will reduce the number of trained parameters in the MLP decoder, and also give compatibility with the diffusion model. The triplane-vector representation is a highly efficient hybrid explicit-implicit network architecture for neural fields. It leverages the concept of projecting a 3D coordinate onto 2D feature planes, $f_{xy}, f_{xz}, f_{yz} \in R^{N \times N \times C}$, where N is the spatial resolution, and C is the number of feature channels, and projecting point on orthogonal to each plane vector embedding $f_x \perp f_{yz}, f_y \perp f_{xz}, f_y \perp f_{xy} \in R^{1 \times 1 \times C}$

(see Figure 3.2). [Chen et al., 2022a, TensoRF: Tensorial Radiance Fields] authors found that triplane-vector decomposition of 3D space is more efficient way to represent high frequency geometry, especially when additional information about multi geometry intersection is needed, than usual triplane parametrization.

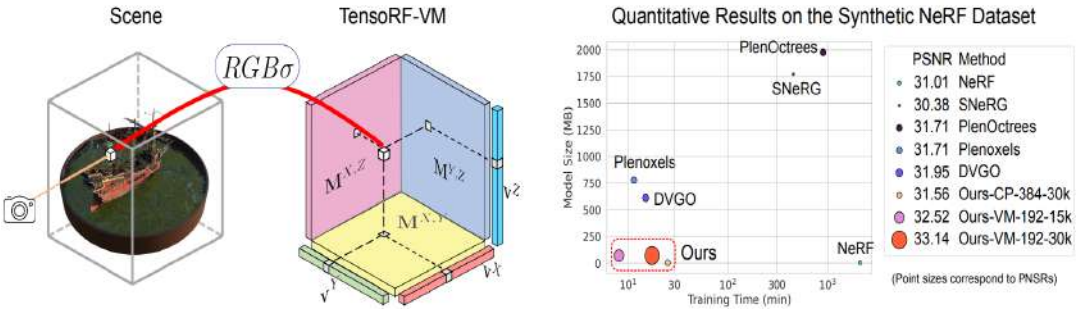

FIGURE 3.2: The authors of "TensoRF: Tensorial Radiance Fields"[Chen et al., 2022a] work showed that their 3D scene parameterization is capable of preserving the scene's high frequency features while keeping the number of parameters required to decode the entire scene small.

Feature Vector Aggregation. For every point in the point cloud, we calculate its projection onto each plane using the corresponding UV coordinates. The feature vectors derived from these projections are subsequently aggregated by summation and passed through a ReLU activation function. The mathematical formulation of this process is as follows:

$$f_i = \text{ReLU}\left(f_{xy}(p_i) \circ f_z(p_i) + fyz(p_i) \circ f_x(p_i) + fxz(p_i) \circ f_y(p_i)\right) \tag{3.1}$$

In this equation, f_i denotes the feature vector for the i-th point p_i and $\text{ReLU}(x) = \max(0, x)$ is the Rectified Linear Unit activation function.

Final Decoding. Implicit MLPs are neural networks trained to learn implicit functions, which define a shape by mapping a 3D point to a scalar value representing the point's relationship to the shape. In our case, the MLP is used to learn a Signed Distance Function (SDF), which provides the shortest distance from a 3D point to the surface of the 3D shape. Positive values indicate the point is outside the shape, while negative values indicate the point is inside, and zero signifies the point is on the surface. Such representation can be formulated as:

$$MLP_\phi(f_i) \rightarrow \hat{d}_i \approx SDF(p_i) \rightarrow d_i \tag{3.2}$$

Where $MLP_\phi(f_i) \rightarrow \hat{d}_i$ represents the output of the neural field for a given point to a distance to closest surface. The feature planes and MLP can be jointly optimized to represent the signed distance function of a shape. This concludes the triplane space parametrization, yielding a compact, efficient, and memory-friendly representation of the 3D object.

Optimization

The optimization process involves several loss functions to guide the learning of the proposed model effectively. The primary goal is to ensure that the predicted Signed Distance Function (SDF) values closely align with the ground truth and that the triplane representation remains spatially smooth.

SDF Loss. The SDF loss function compares the predicted SDF values with the ground truth values. The loss is computed by taking the absolute difference between the predicted and ground truth SDF values. This encourages the model to predict SDF values that closely match the actual distances. The SDF loss can be formulated as follows:

$$\text{SDF Loss} = \frac{1}{N}\sum_{i=1}^{N}|\text{SDF}_{\text{predicted},i} - \text{SDF}_{\text{ground truth},i}| \tag{3.3}$$

Where N is the total number of points in the 3D space, $\text{SDF}_{\text{predicted},i}$ is the predicted SDF value and $\text{SDF}_{\text{ground truth},i}$ is the ground truth SDF value for the i-th point.

Total Variation Loss. Total Variation (TV) loss encourages spatial continuity in the learned triplane representation. It is computed by summing the squared differences between adjacent pixels in the triplane representation along both the height and width dimensions. This regularization discourages abrupt changes in the values of neighboring pixels, promoting a smoother representation.

This process encourages the model to produce accurate SDF predictions while maintaining a spatially smooth triplane representation.

Limitations and Considerations

While the triplane representation technique offers significant advantages, it is not without limitations. The quality of the model highly depends on accurate projection of the 3D positions onto the triplane space, and inaccuracies in this projection may result in an imprecise final 3D representation. Additionally, despite being more memory-efficient than explicit methods, triplane representation still requires considerable computational resources and memory, especially for large or complex objects. Therefore, hardware limitations should be taken into account when working with triplane representation in a practical setting.

3.2 Generative Approach

In this paper, we plan to use the concept of Latent Diffusion Models (LDM) [Rombach et al., 2021] (see Figure 3.3) to synthesize triplane-vector parameterization for implicit space decoder.

3.2.1 Triplane Compression: VAE

Variational AutoEncoders (VAEs) are a pivotal component of the Latent Diffusion Model (LDM) used in our work. These generative models enable us to learn the underlying latent representations of high-dimensional data, such as the triplane representations of 3D meshes.

Overview of Variational AutoEncoders

A VAE comprises an encoder and a decoder. The encoder, or recognition model, maps the high-dimensional input data into a lower-dimensional latent space, outputting a mean and variance. These parameters define a probability distribution from which we can sample in the latent space. The decoder, or generative model, reconstructs the original data from these sampled latent representations.

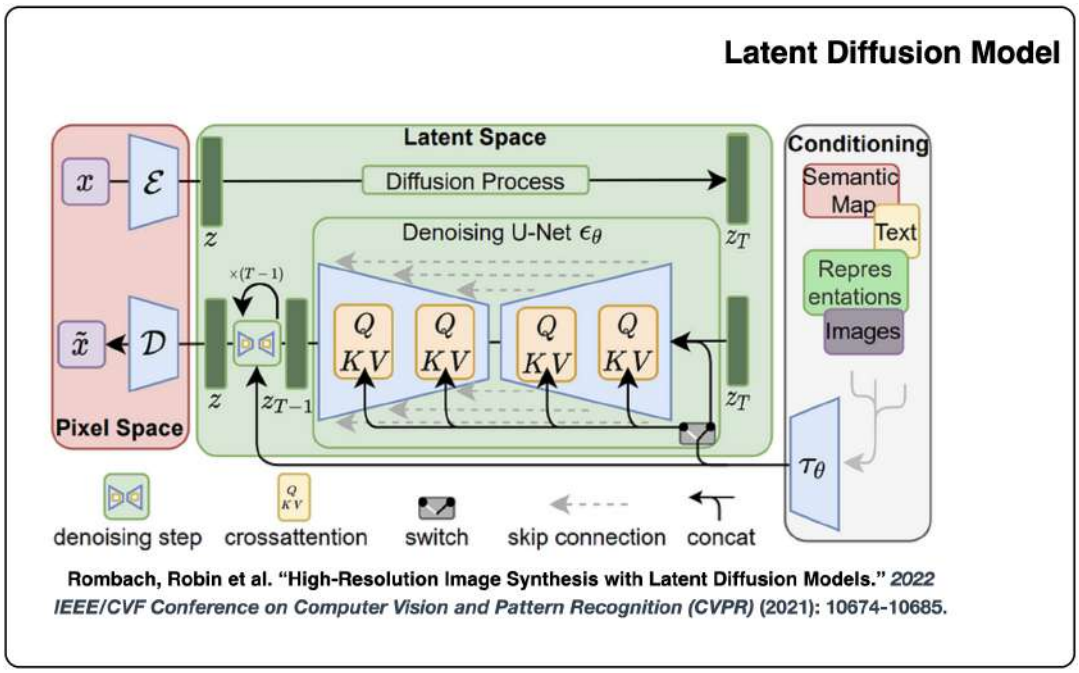

FIGURE 3.3: The authors of "High-Resolution Image Synthesis with Latent Diffusion Models"[Rombach et al., 2021] demonstrated the architecture of their model, showing two important components: a variational autoencoder and a generative model. We plan to use LDM as a triplane texture generator and this is the second component in our text-to-3d pipeline.

In the context of our work, the VAE takes a triplane as input—a 2D representation of the 3D mesh with dimensions $R^{H\times W\times C}$. The VAE further compresses this triplane into a latent representation of dimensions $4 \times 32 \times 32$.

Training the VAE

Training a VAE entails the optimization of two distinct objectives: a reconstruction loss and a regularization loss. The reconstruction loss ensures that the VAE can accurately reconstruct the original data from the latent representations. This is typically evaluated by the $L1$ error between the original and reconstructed data.

The regularization loss, usually expressed as the Kullback-Leibler (KL) divergence, encourages the learned latent distribution to closely align with a predefined prior, often a standard normal distribution. This prevents the model from overfitting to the training data and ensures the generation of diverse samples.

In the context of LDMs, the VAE is optimized in the latent space, enabling it to directly learn the distribution of the data in this space. By efficiently learning to generate triplanes in the latent space, the LDM can then be used to generate novel 3D meshes.

3.2.2 Latent Diffusion Model

Latent Diffusion Models (LDMs) [Rombach et al., 2021] are a significant advancement in the field of generative modeling, particularly for high-resolution image

synthesis. They represent a fusion of Variational Autoencoders (VAEs) and Diffusion Models (DMs), incorporating the strengths of both while mitigating their weaknesses.

Theoretical Background

A Diffusion Model, in its simplest form, is a generative model that learns to generate new data samples by simulating a diffusion process. This process is initialized with a sample from a simple prior distribution (like a standard multivariate Gaussian), and gradually transforms it into a sample from the target distribution through a series of small diffusion steps (see Figure 3.4).

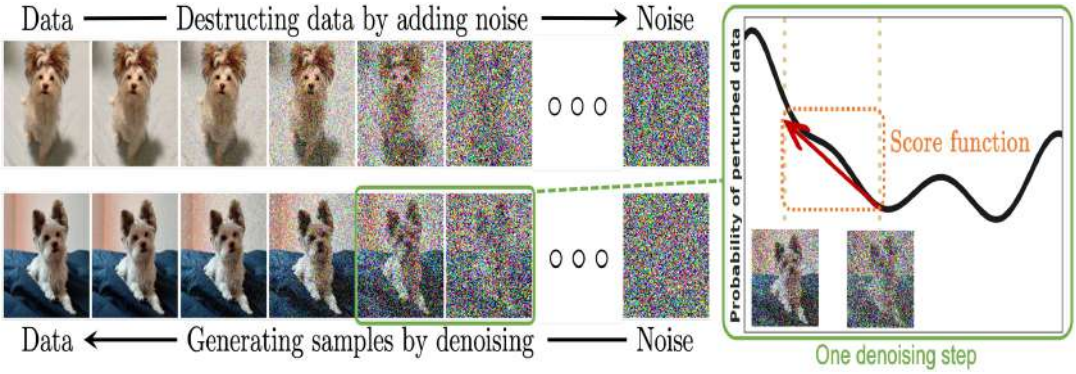

FIGURE 3.4: The authors of "Diffusion Models: A Comprehensive Survey of Methods and Applications"[Yang et al., 2022] work proposed a visualization of how diffusion models iteratively inject noise into data during the forward path, and then learn the reverse process. This visualization helps to understand how diffusion models work and how they can be used to generate new data that is similar to the original data.

The DM's learning process involves two stages. In the first stage, the model transforms a data sample into a simple noise sample with fixed transformation methods. This forward process is typically carried out using fixed Markov Chain Monte Carlo methods. In the second stage, the learned model is used to simulate the reverse diffusion process, which transforms a noise sample into a data sample.

Mathematically, given a data sample x_0 from the target distribution $p(x)$, a diffusion process is applied to generate a sequence of variables $x_0, x_1, ..., x_T$, where x_T is a sample from the prior distribution $p(z)$. The diffusion process is defined by a series of conditional distributions $p(x_t|x_{t-1})$ that satisfies the Markov property. During the learning process, the DM learns to approximate the reverse conditional distributions $q(x_{t-1}|x_t)$.

In the context of LDMs, this process is carried out in the latent space of a VAE, which greatly reduces the computational complexity. The VAE learns a mapping from the data space to a lower-dimensional latent space, and its inverse, through a pair of encoder and decoder networks. The DM then operates on the encoded data in the latent space, instead of the original data in the high-dimensional space.

Mathematical Formulation

Given a triplane x_0, the VAE model downsamples the triplane by a factor d and maps it to a latent space, yielding z_0. The autoencoder is represented as:

$$z_0 = Encoder(x_0) \quad \hat{x}_0 = Decoder(z_0) \tag{3.4}$$

In the case of the diffusion model, we consider a sequence of random variables $Z = z_0, z_1, ..., z_T$ where z_0 is the encoded representation of the original data and z_T is a sample from the prior distribution $p(z)$. This diffusion process is defined by a series of conditional distributions $p(z_t|z_{t-1})$ that satisfy the Markov property. To simulate the reverse process, we learn a set of approximate reverse conditional distributions $q(z_{t-1}|z_t)$ and use them to generate new data samples. The reverse process is represented by the following formula:

$$z_{t-1} = Denoiser(z_t) + \sqrt{1-\beta_t} \cdot \epsilon_{t-1} \tag{3.5}$$

where ϵ_{t-1} is a standard Gaussian noise, β_t is a time-dependent noise level, and Denoiser is a neural network learned to denoise the diffusion process.

The forward process can then be simulated using these learned reverse distributions. The forward process is represented by the following formula:

$$z_t = \sqrt{\beta_t} \cdot z_{t-1} + \sqrt{1-\beta_t} \cdot \epsilon_t \tag{3.6}$$

where ϵ_t is a standard Gaussian noise.

In the end, to generate new data samples, we approximate the reverse process starting from a sample from the prior distribution and decode the final state z_0 back into the data space using the VAE decoder:

$$\hat{x}_0 = Decoder(z_0) \tag{3.7}$$

Benefits and Limitations

LDMs represent a powerful approach to generative modeling, achieving new state-of-the-art results on tasks such as image inpainting, class-conditional image synthesis and various other applications. They offer a near-optimal point between complexity reduction and detail preservation, greatly boosting visual fidelity. Despite these benefits, the training and inference of LDMs can still be computationally expensive, but they are significantly more efficient compared to pixel-based diffusion models. In our work, we propose to utilize this type of model to synthesize triplane textures. We focus on conditioning using class labels from the ShapeNet dataset. Our main contribution is the modification of LDM to shift its focus from generating images to producing triplanes.

From the theoretical side, one can notice an excellent combination of triplane parametrization with latent diffusion models. Triplanes, as a compression of 3D space into 2D, and LDM as a generative method of learning the entire 3D space manifold through 2D proxy triplanes.

It's worth noting that while LDMs can generate high-quality data samples, their performance is still dependent on the quality of the learned VAE model. If the VAE fails to capture the important details of the data, the performance of the LDM will also be affected. As such, careful design and training of the VAE model is crucial for the success of LDMs.

Chapter 4

Training Data

4.1 ShapeNet

In the following section, we present an overview of the ShapeNet [Chang et al., 2015] dataset, which plays a crucial role in our master's thesis. ShapeNet is a comprehensive, large-scale repository of 3D models that covers a wide range of object categories, organized under the WordNet taxonomy. The dataset is richly annotated, with various semantic annotations provided for each 3D model, such as consistent rigid alignments, parts and bilateral symmetry planes, physical sizes, and keywords.

FIGURE 4.1: The authors of the work "ShapeNet: An Information-Rich 3D Model Repository" [Chang et al., 2015] provided an example of what the objects of their dataset look like.

For our research, we specifically focus on the ShapeNet Core subset, which contains a more refined selection of models (see Figure 4.1) from the entire ShapeNet dataset. ShapeNet Core covers 55 object categories with a minimum of 100 models

per category (see Figure 4.2 for categories examples of the complete dataset), offering a cleaner and more uniform dataset for our analysis.

ID	Name	Num	ID	Name	Num	ID	Name	Num
04379243	table	8443	03593526	jar	597	04225987	skateboard	152
02958343	car	7497	02876657	bottle	498	04460130	tower	133
03001627	chair	6778	02871439	bookshelf	466	02942699	camera	113
02691156	airplane	4045	03642806	laptop	460	02801938	basket	113
04256520	sofa	3173	03624134	knife	424	02946921	can	108
04090263	rifle	2373	04468005	train	389	03938244	pillow	96
03636649	lamp	2318	02747177	trash bin	343	03710193	mailbox	94
04530566	watercraft	1939	03790512	motorbike	337	03207941	dishwasher	93
02828884	bench	1816	03948459	pistol	307	04099429	rocket	85
03691459	loudspeaker	1618	03337140	file cabinet	298	02773838	bag	83
02933112	cabinet	1572	02818832	bed	254	02843684	birdhouse	73
03211117	display	1095	03928116	piano	239	03261776	earphone	73
04401088	telephone	1052	04330267	stove	218	03759954	microphone	67
02924116	bus	939	03797390	mug	214	04074963	remote	67
02808440	bathtub	857	02880940	bowl	186	03085013	keyboard	65
03467517	guitar	797	04554684	washer	169	02834778	bicycle	59
03325088	faucet	744	04004475	printer	166	02954340	cap	56
03046257	clock	655	03513137	helmet	162			
03991062	flowerpot	602	03761084	microwaves	152		**Total**	57386

FIGURE 4.2: The authors of the work "ShapeNet: An Information-Rich 3D Model Repository" [Chang et al., 2015] provided statistics on the number of objects for each category from their complete dataset.

We leverage the ShapeNet dataset for pretraining our model, taking advantage of the wealth of 3D geometric information it provides. By pretraining our model on ShapeNet, we aim to develop a robust foundation for our algorithm, enabling it to generalize better when applied to other datasets or real-world scenarios.

Furthermore, we utilize ShapeNet as an official and widely popular benchmark for evaluating the performance of our 3D reconstruction algorithm. By comparing our results with existing state-of-the-art methods on ShapeNet, we can gain insights into the efficacy of our approach and identify areas for potential improvement.

It is important to note that, in our work, we focus solely on the geometric properties of the 3D models and do not consider textures or other visual attributes. This choice allows us to concentrate on the fundamental challenges of 3D shape understanding and reconstruction, without the additional complexities introduced by textural information.

In summary, the ShapeNet dataset, and specifically the ShapeNet Core subset, plays a vital role in our master's thesis. We employ it for pretraining our model, as well as for benchmarking our 3D reconstruction algorithm, emphasizing the geometric aspects of the 3D models while disregarding textures.

4.2 Objaverse

In this section, we will discuss the OBJAVERSE [Deitke et al., 2022] dataset, which serves as the primary data source for our master's thesis. This extensive annotated 3D dataset plays a crucial role in enabling research across various computer vision fields. We will delve into the dataset's statistics, the intended use of its meta information, and the fine-tuning process employed for our text-to-3D model.

The OBJAVERSE 1.0 dataset comprises 818,000 3D objects designed by 160,000 artists (see comparison with ShapeNet in figure 4.3). These objects encompass over

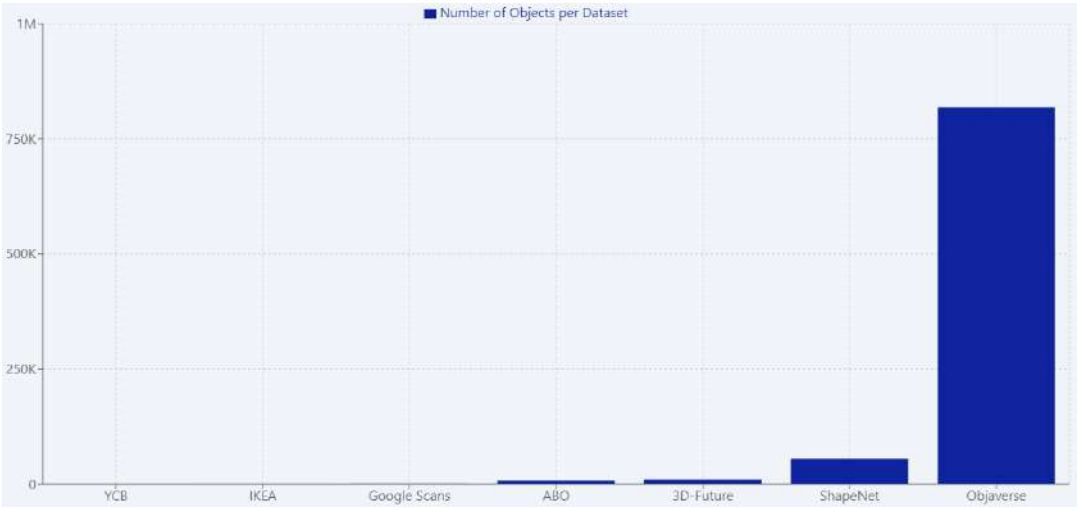

FIGURE 4.3: The authors of "Objaverse: A Universe of Annotated 3D Objects" [Deitke et al., 2022] showed a comparison of their dataset with popular 3D datasets.

2.35 million tags, with more than 170,000 being unique. The dataset is estimated to cover nearly 21,000 WordNet entities. The objects in OBJAVERSE were uploaded between 2012 and 2022, with over 200,000 objects uploaded just in 2021.

Furthermore, OBJAVERSE includes 44,000 animated objects, 63,000 character models, and various articulated objects, exteriors, and interiors. The dataset covers a wide range of visual styles, such as 3D scans, 3D modeled objects, point clouds, and photo-realism through physically based rendering (PBR).

In our research, we propose to leverage the rich meta information of objects from the OBJAVERSE dataset, particularly the text descriptions, to train our text-to-3D model (see Figure 4.4). These descriptions provide valuable insights into the objects' characteristics and can be used as input data for generating accurate 3D models from textual information.

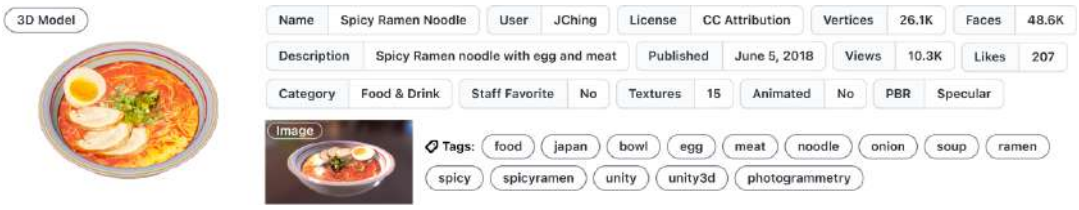

FIGURE 4.4: The authors of "Objaverse: A Universe of Annotated 3D Objects"[Deitke et al., 2022] showed a sample of metadata for each item in OBJAVERSE includes a 3D model, a rendered thumbnail image chosen by the user, a title, a description, tags, a category, statistics, and other supplementary information.

Just like with the shapenet dataset, we are going to only use geometry information, ignoring textures.

In conclusion, the OBJAVERSE dataset serves as a vital resource for our master's thesis, providing us with an extensive collection of annotated 3D objects and metadata. By leveraging this dataset for fine-tuning our text-to-3D model and focusing on geometry, we aim to develop a powerful model capable of generating accurate and detailed 3D representations from textual descriptions.

Chapter 5

Experiments and Implementation Details

5.1 Implementation Details

5.1.1 Data Preprocessing

In this work, we aim to use two datasets for training our models, ShapeNetCore [Chang et al., 2015] and Objaverse [Deitke et al., 2022]. ShapeNetCore is a well-established large-scale dataset of 3D shapes with about 51,000 unique shapes. These shapes are mostly simple, ranging from complex to simple geometries, and contain color information and textures but we will not consider it in our experiments. We plan to split the ShapeNetCore dataset into two parts: one part will be used for pre-training our models and the other for benchmarking as ShapeNetCore is a popular benchmark in the field of 3D synthesis. The details and implementation of the metrics can be found in the relevant section of this paper, see evaluation plan 5.1.3.

On the other hand, Objaverse is a more diverse and rich dataset of 3D figures with descriptions, tags, and animations, often with full textures (with normal, specular, diffuse, refraction, etc. maps) and highly detailed geometry containing over 800K 3D shapes and is updated regularly. In this work, we aim to use only part (100k objects) of the Objaverse dataset as a pretrain for the diffusion model due to the large memory requirements.

For each 3D mesh in our dataset, we performed a two-step sampling process. First, we sampled 5×10^5 points uniformly from the surface of the mesh. Second, an additional 5×10^5 points were sampled uniformly within the spatial extent of the mesh. Each point is represented in $\mathbb{R}^3$.

It is important to highlight that all meshes were normalized to fit within a predefined range. Specifically, the coordinates of all points were scaled such that they fell within the interval $[-1, 1]$. This normalization step ensures consistency across the dataset, facilitating the learning process.

For the initialization of the triplane texture features for each mesh, we adopted a grid structure in $\mathbb{R}^{C \times H \times W}$. Here, C represents the channel size, while H and W denote the height and width of the grid, respectively. We set C to 16, and both H and W were fixed at 256. This structured representation of the triplane texture features enables a systematic and efficient approach to process the 3D data in our experiments.

5.1.2 Training Process

In this study, our objective is to develop a resource-efficient pipeline for text-guided 3D synthesis. We aim to leverage existing, proven architectures as much as possible

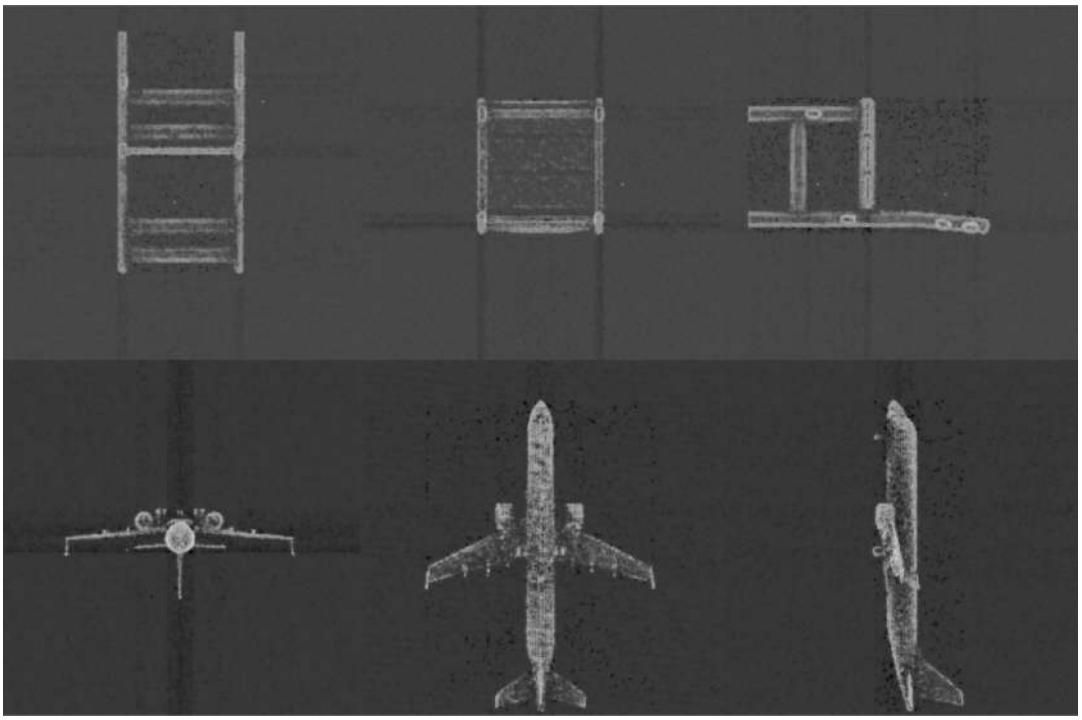

FIGURE 5.1: The illustration herein exhibits an instance of an unrolled texture mapping. For every distinct mesh within the structure, a correspondingly analogous texture is generated through training, which essentially encapsulates pertinent data pertaining to the tri-dimensional representation of the underlying geometry. Furthermore, these tri-dimensional texture mappings possess the capability to encapsulate additional information parameters, including albedo and surface normals.

to expedite our process and harness advancements in the field.

Our proposed solution is a two-stage process. The initial stage involves the random initialization of triplane features for each 3D shape. Here, a triplane feature T for a shape S consists of three orthogonal 2D projections $\{T_{xy}, T_{xz}, T_{yz}\}$ of S on the planes xy, xz, and yz, respectively. Each 2D projection is subsequently encoded into a latent representation using a Multi-Layer Perceptron (MLP) f: $T_{ij} = f(S_{ij})$ for $ij \in \{xy, xz, yz\}$.

In this stage, we train the MLP to refine each triplane feature by backpropagating gradients through an implicit decoder g that shares parameters across all shapes. We utilize the ShapeNetCore dataset to train this stage, with the objective of acquiring a decoder that can accurately reconstruct the point cloud from the triplane features.

In the second stage, we freeze the parameters of the decoder and train the triplane features for all objects from the Objaverse dataset (see Figure 5.1 as an example of triplane texture). This serves to expedite the data processing step.

We also propose a modification to the Latent Diffusion Model (LDM) [Rombach et al., 2021] to extend its capability to texture space. Our approach adheres to a text-to-triplane style, in which the model takes a noisy version of the triplane texture in latent space as input and predicts the noise to be removed.

The training process of the LDM is performed in a classical manner, akin to the approach proposed in the original work [Rombach et al., 2021]. The goal here is to learn a mapping from the noisy triplane textures to their corresponding clean and coherent representations in latent space.

5.1.3 Evaluation Metrics

The evaluation of our proposed model relies on a modified version of the Frechet Inception Distance (FID), a statistical metric designed to measure the dissimilarity between the distribution of generated images and the distribution of real images in the feature space of a pre-trained Inception model. The Inception model, specifically the Inception v3 model, is a convolutional neural network that is widely used in the field of computer vision. It was developed by researchers at Google and has been trained on the ImageNet dataset, which contains over a million images spanning a thousand categories. Due to its high performance on a variety of tasks, it has become a standard model for extracting features from images. Unlike the standard FID, our version considers the rendered shading images of the generated meshes, offering a more thorough and accurate assessment of our model's performance.

This evaluation approach aligns with those used in other works such as [Shue et al., 2022; Zheng et al., 2022]. Notably, the authors in [Zheng et al., 2022] provide a comprehensive exploration and examination of evaluation metrics for 3D generative models, making it a valuable reference for our work.

To ensure a comprehensive evaluation, we follow the procedure outlined in [Shue et al., 2022; Zheng et al., 2022], where 20 distinct views are utilized to render shading images of each generated shape. The FID scores are then computed for each view. These scores are subsequently averaged, leading to the final FID score. The mathematical representation of this procedure is as follows:

$$FID = \frac{1}{20}\left[\sum_{i=1}^{20}\|\mu_g^i - \mu_r^i\|^2 + Tr\left(\Sigma_g^i + \Sigma_r^i - 2\left(\Sigma_r^i\Sigma_g^i\right)^{\frac{1}{2}}\right)\right] \tag{5.1}$$

In the above equation, the variables g and r denote the generated and training datasets, respectively. Additionally, μ^i and Σ^i represent the mean and covariance matrices for the shading images rendered from the i^{th} view.

Our proposed model will be scrutinized under the ShapeNetCore benchmark [Chang et al., 2015], enabling us to compare our solution with other existing solutions [Shue et al., 2022; Nichol et al., 2022; Chou, Bahat, and Heide, 2022; Nam et al., 2022] using the FID metric.

5.2 Experiments

As previously discussed, our pipeline comprises two main components. The first involves the training of triplane textures which are used to parameterize a compact model for point cloud decoding, thus representing the geometry in a three-dimensional space. This portion of the process is the most experimentally intensive within our pipeline. We aim to delve into the crucial experiments that significantly influence the outcomes of our work in this section (see Section 5.2.1). The second component is LDM which we utilize to learn and synthesize triplane textures (see 5.2.2).

All the source code relevant to this research, including algorithms, data analyses, and supplementary scripts, has not been officially released yet. However, early access is available. You can find the repository at this link[1]

[1]For early access, please send a request to daniel.kovalenko@ucu.edu.ua

5.2.1 Triplane Space Parametrization

3D Scene Overfitting

Experiment Setup. At a high level, we have striven to establish a model architecture and training methodology that ultimately yields a 2D representation capable of encapsulating intricate 3D geometries. As such, our experiments broadly fall into categories such as: model architecture (including the number of layers, activations, frequency encoding, and the various types of model output such as SDF, occupancy), triplanes (including optimal size, depth, and methods for sampling and interpolating features), and methods of optimization (including loss functions and their weighting). The primary goal was to balance resource efficiency with the quality of the reconstructions. In other words, the triplane should be compact, the architecture not overly complex, but the overall system should function effectively.

In order to achieve efficient 3D figure reconstruction, we decided to focus only on the light-weighted MLP decoder architecture, this significantly reduces the amount of resources of the entire pipeline, and also makes it possible to render figures in real time using shaders in computer graphics. We are inspired by the use of a simple architecture for the final decoding of a figure by work "MobileNeRF: Exploiting the Polygon Rasterization Pipeline for Efficient Neural Field Rendering on Mobile Architectures" [Chen et al., 2022b], where the authors demonstrate real-time nerf rendering using shaders. Although we do not use NeRF architecture and are focused on the SDF setting, we plan to explore the possibility of online ray marching using our model in shaders.

We initiated our experiments with a straightforward objective: given a single shape or point cloud, we aimed to perfectly reconstruct it using the model weights and information from the triplane - essentially, an overfitting task. This experiment was designed to confirm that the chosen model architecture, triplane, and loss functions were capable of successful pattern reconstruction. As the outcomes of these experiments demonstrated, this is a challenging task, especially considering that an entire subfield of machine learning is devoted to reconstructing the original signal using machine learning as a form of compression (for example, work "Implicit Neural Representations with Periodic Activation Functions"[Sitzmann et al., 2020] shows the complexity of the task of signal reconstruction).

For our architecture comparison experiments, we employed Chamfer loss as the metric. Chamfer loss is a commonly used distance metric for point clouds in the field of machine learning, particularly in tasks concerning 3D reconstruction.

The Chamfer loss between two point clouds, A and B, is defined as:

$$L_{\text{Chamfer}}(A, B) = \frac{1}{|A|} \sum_{a \in A} \min_{b \in B} |a - b|^2 + \frac{1}{|B|} \sum_{b \in B} \min_{a \in A} |a - b|^2 \tag{5.2}$$

Where $|A|$ and $|B|$ denote the number of points in point clouds A and B, respectively. The term $\|a - b\|^2$ denotes the squared Euclidean distance between points a and b.

In simpler terms, the Chamfer loss computes the average minimum squared distance between each point in one point cloud and the closest point in the other point cloud. This calculation is performed in both directions, from A to B and from B to A, to account for discrepancies in both point clouds. This makes it a symmetric loss function that measures the similarity between the two point clouds. Minimizing this loss during training makes the generated point cloud increasingly similar to the target point cloud.

Results. The performance of the proposed text-in-3D model is evaluated based on two primary aspects: the loss curve and the optimal hyperparameters.

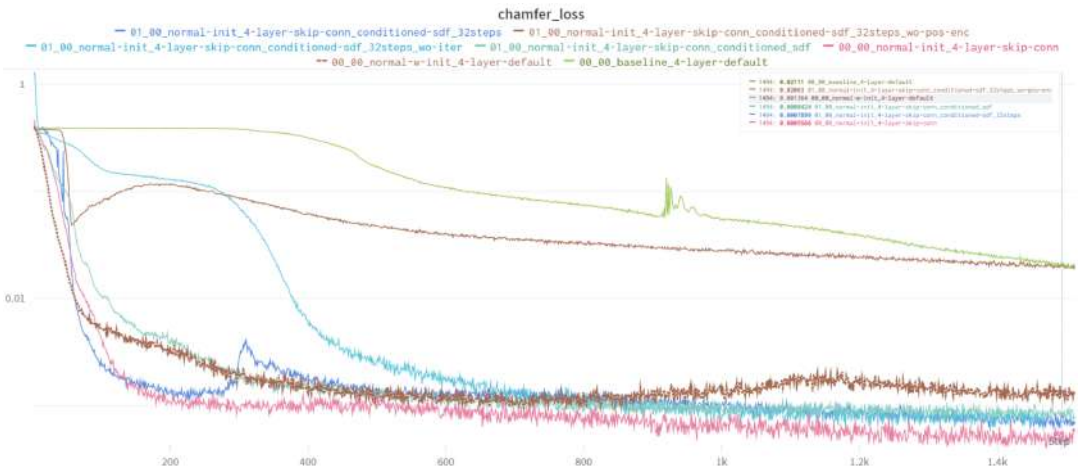

FIGURE 5.2: This figure shows the loss charts of different experiments. Chamfer loss was used as a metric. In more detail: the most optimal model is a four-layer MLP, with a skip-connection block in the hidden layer, which has 128 hidden size, as well as with frequency encoding of the input features.

Figure 5.2 depicts the loss curves during the training phase of our model. It can be observed that the loss decreases steadily over iterations, indicating that the model is learning effectively. However, the loss curve does not reach a minimal value, suggesting room for potential improvements.

Parameter Name	**Searchable values**
Hidden Size	32, 64, **128**, 256
N Layers	2, 3, **4**
PointCloud Initialization	**Uniformal**, Normal
Larning Rate	**0.001**, 0.0001, 0.00001
triplane Feature Blending Methods	**Concat**, Sumup, Multiplication
Frequency Encoding	**Yes**, No
Activation Type	**ReLu**, Sine, Sigmoid, ReLu + Sigmoid

TABLE 5.1: Hyperparameter settings for MLP experiments. The best performing set (Experiment 3) is highlighted in bold.

Table 5.1 shows the optimal hyperparameters determined through an experimental search process. Hyperparameters were selected manually inspired by work [Shue et al., 2022], and a random search of parameters was also used. These bold parameters provided the best performance for our model on the given one figure overfitting task. This study cannot be called complete. As the authors of the work "Implicit Neural Representations with Periodic Activation Functions"[Sitzmann et al., 2020] show, signal reconstruction is not an easy task. One of the potential areas of research could be the study of periodic activation functions, however, in such a scenario, a large number of experiments will appear to find the optimal initialization of the model (since weights initialization is the crucial for such activations), as well as its regularization to achieve generalization.

Pointcloud-to-triplane encoding

In the process of developing a pipeline capable of generating a dataset of meshes and their corresponding triplanes, we relied on two works [Gupta and Gupta, 2023; Shue et al., 2022]. The first [Gupta and Gupta, 2023] proposed the use of a 3D Variational Autoencoder (VAE) model. This model comprises two main components: an encoder, which utilizes PointNet++[Qi et al., 2017] for conditioning a UNet-like model, and a decoder, which employs the Marching Tetrahedra algorithm as in [Gao et al., 2022].

Our primary focus was on the encoder, as we used an implicit signed distance function (SDF) model for our decoder. However, after numerous trials, the results were not satisfactory; the model either failed to converge or demonstrated poor convergence. Notably, when utilizing periodic activation functions such as sine, the model failed to converge entirely. This led us to surmise that the poor convergence could be attributed to the gradient flow of Siren-like models. Another drawback of such an architecture is its high computational cost due to the complexity of the UNet architecture, making it extremely expensive to train.

Consequently, we decided to pivot to a different strategy, namely, the generation of triplanes through the propagation of gradients in an implicit SDF. This method is elaborately discussed in the paper "3D Neural Field Generation using Triplane Diffusion" [Shue et al., 2022]. The underlying concept is similar to the overfitting experiment, but in this case, we iterate over the meshes, initializing different triplanes for each. The decoder weights remain common across the iterations.

In an effort to enhance the quality of the textures, we incorporated regularization techniques, specifically total variance loss and L_2 regularization.

Results. Following our experimentation, we successfully generated approximately 50,000 triplane textures for the ShapeNet dataset, and around 100,000 textures for the Objaverse dataset. The major challenge encountered during this experiment was the storage of these textures, given that a single texture occupies 256x256x48 float values. This necessitated the development of extensive infrastructure for file compression. Despite these efforts, the dataset still occupies terabyte of storage space and this is even to a full dataset.

5.2.2 Training of the Variational Autoencoder and Diffusion Model

VAE training. Our perceptual compression model is based on a work "High-Resolution Image Synthesis with Latent Diffusion Models" [Rombach et al., 2021] and consists of a variational autoencoder trained by a combination of a perceptual loss and a patch-based adversarial objective. This ensures that the reconstructions are confined to the triplane manifold by enforcing local realism and avoiding blurriness introduced by relying solely on feature-space losses.

The configurations of the VAE and Diffusion model were kept largely unchanged. The minor modifications made were primarily for adapting shapes from images to triplanes. The VAE was trained over a period of two weeks using four T4 graphics cards. The compression achieved was approximately eightfold, while the quality of the reconstruction remained optimal (see Figure 5.3).

The objective of the training process was to optimize the parameters of the model to minimize the loss function, which is a combination of the Negative Log-Likelihood (NLL) loss and the Kullback-Leibler (KL) divergence. The NLL loss measures the capability of the VAE to reconstruct the input data, whereas the KL divergence imposes

FIGURE 5.3: In this image, you can see two unwrapped triple textures. Above is a ground truth texture obtained from a random object from the ShapeNetCore dataset. Below is a reconstruction. As you can see, they are quite difficult to distinguish.

a regularization effect on the latent space by comparing the learned latent distribution to a prior distribution. During the training, the total loss showed a consistent decrease, indicating that the VAE was progressively improving its ability to reconstruct the input data. This trend is illustrated in Figure 5.4.

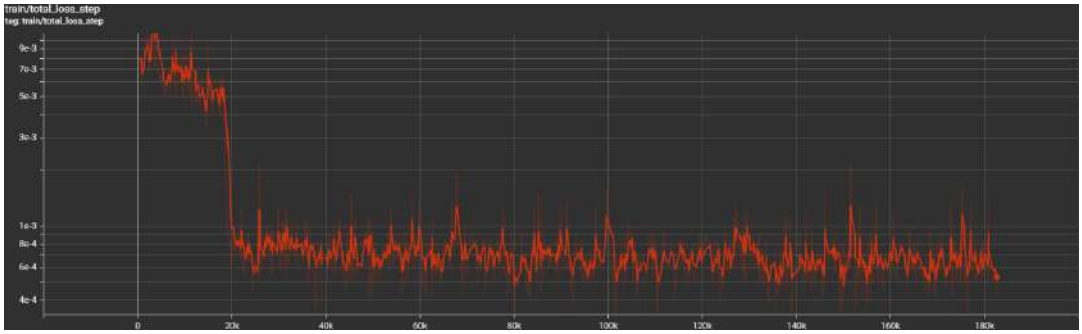

FIGURE 5.4: Convergence of total loss during the training of the VAE.

The decrease in the total loss suggests that the model was effectively learning the underlying structure of the data, providing a good fit to the data distribution and increasing its capacity to generate plausible samples.

Diffusion Model training. The base configuration of the diffusion model underwent minor modifications, primarily the adaptation from pixel space to triplane space. This shift in representation allowed the model to better capture the intricacies of the data it was trained on.

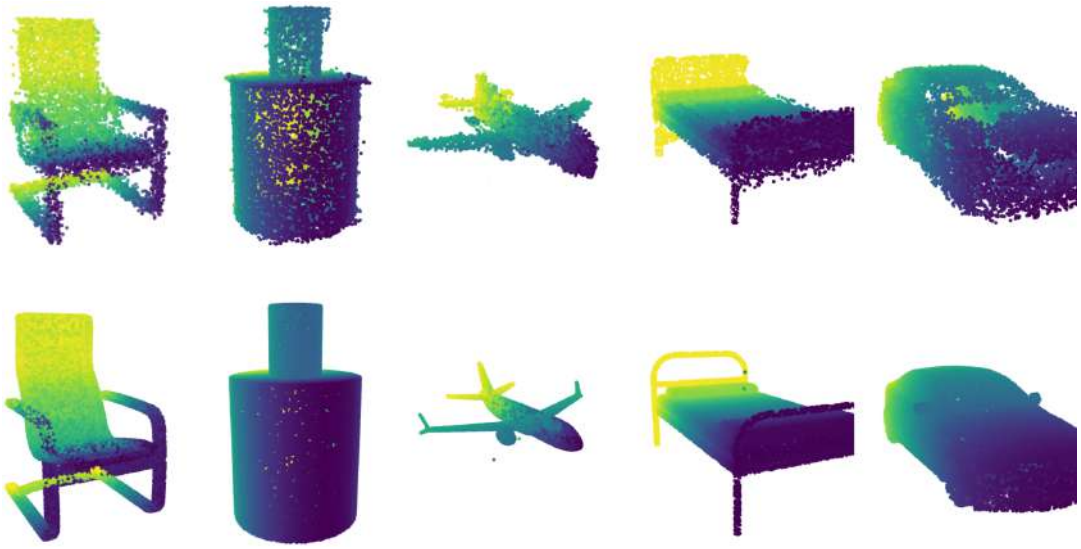

FIGURE 5.5: An example of how our model generates 3D figures in an unconditional manner.

Visualizations provided offer a clear depiction of the model's training progression. The first visualization demonstrates examples of diffusion unconditional generations (see Figure 5.5), showcasing the model's ability to generate consistent outputs. The second visualization provides a row of denoising steps for an image, offering insight into the gradual transformation and improvement the model undertakes during the denoising process (see Figure 5.6). Our diffusion model operates within the VAE's latent space. As previously mentioned, the size of the latent representation for a triplane is $32 \times 32 \times 4$.

The diffusion model was trained over the course of approximately two weeks on 4 T4 video cards. Throughout this training period, the model's loss continued to decrease, signaling the continuous learning and improvement of the model. This consistent decrease in loss is indicative of the model's convergence, a phenomenon wherein the evolution of the model stabilizes as it approaches an optimal state. It's worth noting, however, that while convergence is often a desired outcome, premature convergence can lead to suboptimal results, and it's vital to ensure that the model doesn't stagnate at a less than ideal state.

The total loss of diffusion model still falling, so there is potential to continue experiments (see Figure 5.7).

It's worth noting that the architecture of this model inherently supports text guidance through the use of CLIP embeddings. However, this entails additional training that is resource-intensive. As such, the current model checkpoint does not support text guidance, but it can be easily adapted to do so. Through this work, our objective is to demonstrate the potential of this architecture, and we plan to adapt it for text guidance in future iterations.

5.2.3 Results

We conducted a multitude of experiments, which can be broadly categorized into two distinct types, mirroring the global division of the pipeline itself.

In the first category, we carried out a series of tests aiming to compress 3D space into a 2D feature grid, termed as a triplane. The directions of experiments included:

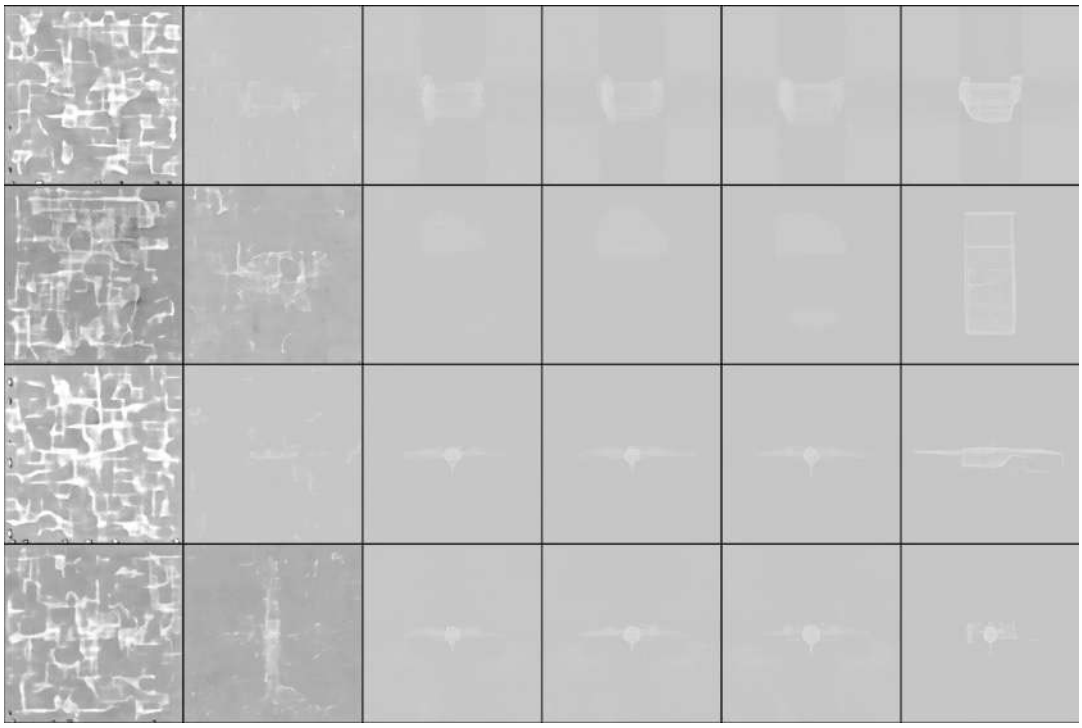

FIGURE 5.6: This is an image of one plane out of three, where the batch size is 4. Here you can see how the model starts generation from a Gaussian distribution and sequentially reaches recognizable patterns of 3d shapes. This model is still converging and the results are not final.

the model architecture; triplane parameters; and optimization methods. Our primary objective was to strike a balance between resource efficiency and the quality of reconstructions. In essence, we aimed for a compact triplane, a simple but effective architecture, and an overall efficient system (see Section 5.2.1 for more details).

In the second category of experiments, we adapted the latent diffusion model to triplanes. We found that the model converges satisfactorily even in its basic configuration (see Section 5.2.2 for more details).

To evaluate our model and compare it to other works, we employed the Frechet Inception Distance (FID) metric. We generated 5,000 meshes for each category in a label conditioned manner, after which we computed the FID, as described earlier in Section 5.1.3.

The performance of our algorithm is observed to be considerably inferior to that of contemporary state-of-the-art models (see Table 5.2). We conjecture that this deficiency originates from error accumulation during data processing. A more thorough investigation is needed to optimize the sampling process from 3D geometries, as well as to refine how the geometry signal is encoded into triplines. Another factor worth scrutinizing is the possibility that the diffusion model has not reached convergence.

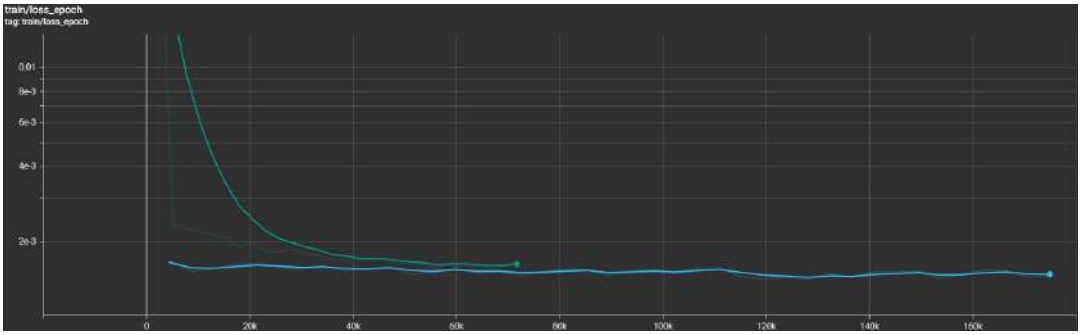

FIGURE 5.7: Convergence of total loss during the training of the diffusion model. The model was trained on the Objaverse dataset, represented in green, and subsequently on the ShapeNet dataset, represented in blue.

Data	Method	FID ↓
Cars	PVD*	335.8
	Implicit-Grid	209.3
	Ours	183.43
	SDF-StyleGAN	98.0
	NFD	83.6
Chairs	PVD*	305.8
	Implicit-Grid	119.5
	Ours	68.05
	SDF-StyleGAN	36.5
	NFD	26.4
Planes	PVD*	244.4
	Implicit-Grid	145.4
	Ours	127.19
	SDF-StyleGAN	65.8
	NFD	32.4

TABLE 5.2: The FID of our model is better than some baselines, but it is still far from being a state-of-the-art model.

Chapter 6

Conclusions

6.1 Contribution

In this study, we have introduced a potential solution to the text-to-3D generative problem.

A comprehensive analysis of existing methodologies in the field was conducted, which allowed us to identify three distinct categories: diffusion models as a prior, diffusion models over 3D compression, and 3D diffusion models.

Our proposed approach strives to find a balance between speed and memory requirements by implementing 2D diffusion over a memory-intensive compressed 3D representation such as a triplane. This representation effectively reduces the problem dimensionality to 2D and allows a simpler data distribution manifold due to one dimension's compression, consequently facilitating high-quality generations.

We are among the pioneers to suggest the use of the Objaverse dataset, a rich resource with over 800k objects. The dataset offers not only high-quality meshes but also significant semantic context via textual descriptions.

Our model exhibits versatility in generating content in both conditioned and unconditioned manners. The two-stage training process and the availability of a dataset of triplane textures pave the way for experimentation with diffusion models, their modifications, and conditioning modifications. We have utilized class labels from the ShapeNet dataset, but the inclusion of CLIP embeddings remains a viable alternative.

In conclusion, our experiments led to the creation of a generative model that, despite not matching state-of-the-art metrics, demonstrated the advantages of a 3D compression approach referred to as triplane. Our model exhibits the capability to generate 3D content (which inherently presents a more complex structure than 2D) in roughly 45 seconds on T4 video cards.

6.2 Future Steps

A critical subsequent step is to modify the current checkpoint so that it supports text-guided generation instead of class-label guidance. Fortunately, this transition does not necessitate any alterations to the existing structure or the current checkpoint. This endeavor will involve fine-tuning the current checkpoint using the comprehensive Objaverse dataset. Various experiments were undertaken in this study, encompassing different data types (such as SDF and occupancy fields) and diverse architectures. Our findings underscored the critical role played by the quality of triplanes. This parameterization displays encouraging outcomes and is not fully realized its potential. The potential use of 3D VAE (Vatiational Autoencoder) for embedding 3D space into a shared latent space of a single model warrants thorough consideration.

This is pivotal as it ensures that features across all triplanes are derived from the same prior distribution, as each triplane is presently fitted independently, leading to ambiguity when fitting the decoder and diffusion.

Incorporating color information and normals could be advantageous, albeit complicating the task due to the additional data dimension, it does provide an auxiliary geometry information source to the model.

It would be beneficial to train the model on the entire Objaverse dataset, as currently, only a small fraction is utilized for pre-training our diffusion model.

Lastly, investigating methods for converting point clouds into a mesh for subsequent export and utilization in the computer graphics industry is an essential aspect of this study.

Bibliography

Barron, Jonathan T. et al. (2021). "Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields". In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5835–5844.

Chang, Angel X. et al. (2015). *ShapeNet: An Information-Rich 3D Model Repository*. Tech. rep. arXiv:1512.03012 [cs.GR]. Stanford University — Princeton University — Toyota Technological Institute at Chicago.

Chen, Anpei et al. (2022a). "TensoRF: Tensorial Radiance Fields". In: *European Conference on Computer Vision*.

Chen, Zhiqin et al. (2022b). "MobileNeRF: Exploiting the Polygon Rasterization Pipeline for Efficient Neural Field Rendering on Mobile Architectures". In: *ArXiv* abs/2208.00277.

Chou, Gene, Yuval Bahat, and Felix Heide (2022). "DiffusionSDF: Conditional Generative Modeling of Signed Distance Functions". In: *ArXiv* abs/2211.13757.

Deitke, Matt et al. (2022). "Objaverse: A Universe of Annotated 3D Objects". In: *ArXiv* abs/2212.08051.

Dhariwal, Prafulla and Alex Nichol (2021). "Diffusion Models Beat GANs on Image Synthesis". In: *ArXiv* abs/2105.05233.

Gao, Jun et al. (2022). "GET3D: A Generative Model of High Quality 3D Textured Shapes Learned from Images". In: *ArXiv* abs/2209.11163.

Goodfellow, Ian J. et al. (2014). "Generative Adversarial Nets". In: *NIPS*.

Gupta, Anchit and Anchit Gupta (2023). "3DGen: Triplane Latent Diffusion for Textured Mesh Generation". In: *ArXiv* abs/2303.05371.

Ho, Jonathan, Ajay Jain, and P. Abbeel (2020). "Denoising Diffusion Probabilistic Models". In: *ArXiv* abs/2006.11239.

Ho, Jonathan et al. (2022). "Imagen Video: High Definition Video Generation with Diffusion Models". In: *ArXiv* abs/2210.02303.

Karnewar, Animesh et al. (2022). "ReLU Fields: The Little Non-linearity That Could". In: *ACM SIGGRAPH 2022 Conference Proceedings*.

Karras, Tero et al. (2019). "Analyzing and Improving the Image Quality of StyleGAN". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8107–8116.

Lin, Chen-Hsuan et al. (2022). "Magic3D: High-Resolution Text-to-3D Content Creation". In: *ArXiv* abs/2211.10440.

Mildenhall, Ben et al. (2020). "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis". In: *European Conference on Computer Vision*.

Mirza, Mehdi and Simon Osindero (2014). "Conditional Generative Adversarial Nets". In: *ArXiv* abs/1411.1784.

Müller, Thomas et al. (2022). "Instant neural graphics primitives with a multiresolution hash encoding". In: *ACM Transactions on Graphics (TOG)* 41, pp. 1 –15.

Nam, Gimin et al. (2022). "3D-LDM: Neural Implicit 3D Shape Generation with Latent Diffusion Models". In: *ArXiv* abs/2212.00842.

Nichol, Alex and Prafulla Dhariwal (2021). "Improved Denoising Diffusion Probabilistic Models". In: *ArXiv* abs/2102.09672.

Nichol, Alex et al. (2022). "Point-E: A System for Generating 3D Point Clouds from Complex Prompts". In: *ArXiv* abs/2212.08751.

Poole, Ben et al. (2022). "DreamFusion: Text-to-3D using 2D Diffusion". In: *ArXiv* abs/2209.14988.

Qi, C. et al. (2017). "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space". In: *NIPS*.

Radford, Alec et al. (2021). "Learning Transferable Visual Models From Natural Language Supervision". In: *International Conference on Machine Learning*.

Ramesh, Aditya et al. (2022). "Hierarchical Text-Conditional Image Generation with CLIP Latents". In: *ArXiv* abs/2204.06125.

Razavi, Ali, Aäron van den Oord, and Oriol Vinyals (2019). "Generating Diverse High-Fidelity Images with VQ-VAE-2". In: *ArXiv* abs/1906.00446.

Rombach, Robin et al. (2021). "High-Resolution Image Synthesis with Latent Diffusion Models". In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10674–10685.

Saharia, Chitwan et al. (2022). "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding". In: *ArXiv* abs/2205.11487.

Schneider, Flavio, Zhijing Jin, and Bernhard Scholkopf (2023). "Moûsai: Text-to-Music Generation with Long-Context Latent Diffusion". In: *ArXiv* abs/2301.11757.

Schuhmann, Christoph et al. (2022). "LAION-5B: An open large-scale dataset for training next generation image-text models". In: *ArXiv* abs/2210.08402.

Shue, Jessica et al. (2022). "3D Neural Field Generation using Triplane Diffusion". In: *ArXiv* abs/2211.16677.

Sitzmann, Vincent et al. (2020). "Implicit Neural Representations with Periodic Activation Functions". In: *ArXiv* abs/2006.09661.

Sitzmann, Vincent et al. (2021). "Light Field Networks: Neural Scene Representations with Single-Evaluation Rendering". In: *Neural Information Processing Systems*.

Song, Jiaming, Chenlin Meng, and Stefano Ermon (2020). "Denoising Diffusion Implicit Models". In: *ArXiv* abs/2010.02502.

Suhail, M. Mohamed et al. (2021). "Light Field Neural Rendering". In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8259–8269.

Wang, Tengfei et al. (2022). "Rodin: A Generative Model for Sculpting 3D Digital Avatars Using Diffusion". In: *ArXiv* abs/2212.06135.

Wu, Rundi et al. (2023). "Sin3DM: Learning a Diffusion Model from a Single 3D Textured Shape". In.

Yang, Ling et al. (2022). "Diffusion Models: A Comprehensive Survey of Methods and Applications". In: *ArXiv* abs/2209.00796.

Zeng, Xiaohui et al. (2022). "LION: Latent Point Diffusion Models for 3D Shape Generation". In: *ArXiv* abs/2210.06978.

Zhang, Biao et al. (2023). "3DShape2VecSet: A 3D Shape Representation for Neural Fields and Generative Diffusion Models". In: *ArXiv* abs/2301.11445.

Zheng, Xin et al. (2022). "SDF-StyleGAN: Implicit SDF-Based StyleGAN for 3D Shape Generation". In: *Computer Graphics Forum* 41.