CLIP-MESH: GENERATING TEXTURED MESHES FROM TEXT USING PRETRAINED IMAGE-TEXT MODELS

Nasir Mohammad Khalid

A thesis
in
The Department
of
Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Science (Computer Science) at
Concordia University
Montréal, Québec, Canada

April 2023

Concordia University
School of Graduate Studies

This is to certify that the thesis prepared

By: Nasir Mohammad Khalid

Entitled: **CLIP-Mesh: Generating textured meshes from text using pretrained image-text models**

and submitted in partial fulfillment of the requirements for the degree of

Master of Science (Computer Science)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

Dr. Yiming Xiao — Chair

Dr. Yang Wang — Examiner

Dr. Eugene Belilovsky — Co-supervisor

Dr. Tiberiu Popa — Supervisor

Approved by

Dr. Lata Narayanan, Chair
Department of Computer Science and Software Engineering

Approved by

Dr. Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

CLIP-Mesh: Generating textured meshes from text using pretrained image-text models

Nasir Mohammad Khalid

The following thesis introduces a novel technique for generating textured mesh models without any 3D supervision based solely on a text prompt. This is done by deforming the control shape of a limit subdivided surface along with its texture and normal map to match an input text prompt. The generated mesh asset can be easily integrated into games or modeling applications that rely on widespread rasterization based rendering techniques. The approach relies on a pre-trained Contrastive Language-Image Pre-Training (CLIP) model to compare the input text prompt with differentiably rendered images of our initialized 3D model. Unlike previous works that focused on stylization or required training of generative models, it performs optimization on mesh parameters directly to generate shape, texture, or both. To ensure that the optimization produces plausible meshes and textures, this work introduces several techniques including image augmentations, camera tuning and use of a pre-trained prior that generates CLIP image embeddings given a text embedding. Overall, this method offers a promising solution for zero-shot generation of 3D models, demonstrating the potential of CLIP-based techniques for the field of computer graphics.

Acknowledgments

I would like to express my deepest gratitude to my parents, whose unconditional love and unwavering support have been a constant source of strength throughout my academic journey. I am also grateful to my brothers, who have always been there for me, providing encouragement and inspiration. To my beloved wife Misha, thank you for your patience, understanding, and for believing in me even when I doubted myself.

I would also like to extend my sincere appreciation to my academic advisors, Dr. Eugene Belilovsky and Dr. Tiberiu Popa, for their invaluable guidance, wisdom, and mentorship. Their dedication to my academic success and their willingness to go above and beyond in their support of my research has been instrumental in shaping the direction of this thesis. I am truly fortunate to have had such outstanding advisors, and I will always be grateful for their contributions to my intellectual growth.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

3D models play a crucial role in various fields such as gaming, virtual reality, film, architecture, and product design. They enable us to create digital representations of objects, scenes, and environments that closely resemble their real-life counterparts, allowing us to study, analyze, and communicate their features and properties in a virtual environment. In gaming, these models are used to create characters, props, and environments that are central to gameplay, immersing the player in the game world. In architecture and product design, they help designers and architects to visualize and iterate on designs before actual production, saving time and resources. In medical imaging, models enable doctors and researchers to study anatomy and simulate surgical procedures, improving patient outcomes. The importance of 3D models lies in their ability to facilitate efficient and effective visualization, design, and communication in a wide range of industries and applications.

The creation of 3D models can be accomplished using a variety of techniques, including volumetric representations, point clouds, and parametric surfaces. However, the polygonal mesh is the most commonly used format in games and other real-time applications due to its efficiency and versatility. Meshes consist of a collection of vertices, edges, and faces that define the shape of the object, which can be rendered using a variety of shading techniques and texture maps. Additionally, meshes can be easily transformed and manipulated through various techniques, such as subdivision, deformation, and sculpting. This flexibility and efficiency make meshes an ideal choice for real-time applications, where computational resources are limited, and performance is critical.

The demand for 3D models is on the rise and is expected to continue growing in the

future. This is due to several factors, including advancements in technology, increased accessibility, and a growing number of applications. The rise of virtual and augmented reality technologies has led to a demand for large number of models to create immersive experiences. Furthermore, multiple industries such as entertainment, gaming, and advertising has also contributed to the growing demand for 3D models, as these technologies and industries continue to develop, it is likely that the demand will continue to grow.

To meet this demand, there is a need to introduce more efficient and accurate 3D mesh generation techniques. Traditional techniques such as modelling and even procedural generation rely on some manual input and require a significant amount of time and effort to setup. Newly introduced deep learning-based 3D mesh generation techniques rely on neural networks that are trained on large datasets of 3D models to generate new meshes. These networks are able to learn the underlying patterns and structures of 3D models, allowing them to generate high-quality meshes that are accurate and efficient.

Deep learning-based 3D mesh generation techniques have several advantages over traditional techniques. They are scalable, flexible, and can generate high-quality meshes quickly and efficiently. Additionally, they can be trained to generate meshes that are specific to certain applications or industries, making them more versatile than traditional techniques.

However, they rely on large scale 3D shape datasets which are restricted in terms of size, quality and diversity. Existing datasets cannot allow for varied shape generation and since creating such a dataset is cost prohibitive we look at alternative techniques such as leveraging the knowledge of existing multimodal models and attempting to exploit their image generative capabilities to create 3D meshes and textures

1.2 Main Contributions

Our main contributions can be summarized as follows:

- We introduce a set of techniques that allow zero-shot text guided generation with a differentiable renderer.
- We use these techniques to directly generate 3D meshes with their texture maps and normal maps.
- We use the analytical expression of the Loop subdivision limit surface as an implicit regularizer to improve the quality of the generated model.

- We improve on our baseline results by introducing a set of render augmentations and incorporating a text to image embedding prior.
- The result of this work has been published in SIGGRAPH Asia 2022 [9]

1.3 Thesis Outline

The next Chapter 2 discusses prerequisite information that makes up the core of the papers methodology. The following Chapter 3 discusses other research and work in the field of generative deep learning and briefly discusses how our work relates to them. We then present the main technique used for generating shapes in Chapter 4. Chapter 5 shows results from our proposed technique and also explores the versatility of the technique by showing results from various different modeling scenarios. We then compare our work quantitatively to related research and perform an ablation study in Chapter 6. Finally we conclude the thesis in Chapter 7 with a short discussion on future works and contributions.

Chapter 2

Background

In this chapter, we explain briefly the background materials related to the works presented in this manuscript.

2.1 Mesh Generation

Polygonal meshes are typically created using specialized 3D modeling software, where the vertices, edges, and faces of the mesh are manipulated to create the desired object [10]. This process can be both challenging and labor-intensive, requiring significant time and effort to achieve the desired level of detail and accuracy. The texturing of meshes can be an equally challenging task, requiring expertise in digital painting and UV mapping.

In order to meet the growing demand mentioned in the previous section, researchers and industries have turned to automatic mesh generation techniques. Automatic mesh generation involves using algorithms to generate a 3D mesh based on certain input parameters, such as the text description, primitive templates or other modalties such as 2D images.

Research in automatic mesh generation has focused on developing algorithms that can generate meshes with high levels of detail and accuracy. One approach is to use machine learning techniques, such as deep neural networks, to learn the mapping between 2D images and 3D meshes [11,12]. Further methods are discussed in Chapter 3, but one of the biggest challenges amongst all research is the lack of large and diverse datasets of 3D objects. This limitation has made it difficult to train and evaluate 3D generative models using real-world data.

One of the most commonly used 3D object datasets is Shapenet [13]. The dataset contains 55,000 3D models from 50 object categories, and each object is represented

by a 3D mesh. While this dataset has been widely used in research, it is limited in terms of its object variety and natural language descriptions. Similarly, CO3D [14] is another 3D object dataset that contains 50 object categories with a total of 8,000 objects. While this dataset includes natural language descriptions of the objects, it is still relatively small compared to large-scale 2D image datasets.

On the other hand, large-scale 2D image datasets such as ImageNet-21K [15] have been critical to the development of computer vision algorithms. ImageNet-21K contains 14 million images of 21,000 object categories and has been used to train deep neural networks for a wide range of computer vision tasks. In addition to the large number of images, it also includes textual descriptions of each object, which can be used to train models for object recognition and understanding.

These large image datasets have led to breakthroughs in image generation tasks [3,16] in terms of quality and scope of generation but the same cannot be said for 3D generative works with the lack of similar datasets being the key driving factor

2.2 Multimodal Deep Learning

Multimodal deep learning is a subfield of deep learning that deals with the processing of data from multiple modalities [17]. This includes various forms but most commonly relates to text, images, audio and video. It aims to combine the information from all these sources to learn a better representation of the underlying data, leading to improved performance in tasks such as classification, prediction and generation. [3,18]

One of the key challenges is the integration of data from different modalities. Since each modality has its own features and characteristics, it is essential to design models that can effectively combine them. This can be achieved through various neural network architectures that learn to effectively extract and combine features from multiple sources, such as using convolutional neural network layers to process images and recurrent neural networks to process text [19]. Multimodal training also needs large and diverse datasets which need to contain information from multiple modalities that are relevantly connected and labelled.

2.2.1 Contrastive Language–Image Pre-training

CLIP (Contrastive Language–Image Pre-training) [20] is a multimodal neural network that was introduced by OpenAI, the authors of the work argue that existing methods for training visual models are limited by the availability of annotated data and

require expensive and time-consuming annotation processes. Furthermore, they cite the fixed number of classes in datasets as being a limitation to unlocking full scale image classification and recognition. Instead, they propose using natural language descriptions of images as a form of supervision, allowing for more flexible and scalable training.

The authors use a private dataset of images scraped from the internet along with their alt-text descriptions to train a two part model: The first being a vision transformer [21] based image encoder that maps an image to a fixed size latent space, the second being a transformer [22] based text encoder that maps text to the same latent space. The two models are trained simultaneously using a contrastive learning framework, which involves learning to discriminate between positive and negative image-caption pairs. This approach encourages the model to learn representations that capture the underlying semantic and visual relationships between the images and their associated text.

The final result is a powerful model that can provide a score between text and images. It improves the performance of the model on a variety of downstream visual and language tasks, including image captioning, visual question answering, and text-based image retrieval. The method is also flexible and can be adapted to different types of image and text datasets, making it a valuable tool for a wide range of applications. One of the key benefits of this approach is its ability to learn transferable visual and language representations. This means that the representations learned by the model can be applied to new datasets and tasks without the need for additional pre-training.

2.3 Differentiable Rendering

Rendering is a well documented and exhaustively studied topic of computer graphics research. However, with the recent deep learning revolution there has been a revival of the fundamental rendering research examined through the lens of backpropogation and gradient descent. One of the main advantages of differentiable rendering is that it allows for end-to-end optimization of a rendering pipeline, which means that the entire rendering process can be optimized using gradient descent and backpropagation. This means that any part of the rendering pipeline can be replaced with optimizable parameters or neural networks.

The use of differentiable rendering has led to state of the art results in tasks such as single view 3D reconstruction tasks [23, 24] and human pose estimation [25] because

the underlying shapes can be rendered to images and now be optimized through image based losses with backpropogation. Additionally, new algorithms for shape representation and image synthesis have also been enabled such as neural radiance fields (NeRFs) [26] and neural implicit volumes [27]. These algorithms use neural networks to learn the implicit representation of a 3D scene from a set of input images, which can then be used to synthesize new images from different viewpoints or be used as high resolution shape representations. These implicit representations are only possible due to the ability to render and optimize them using gradient based optimization

2.4 Limit Subdivision

Loop subdivision [28] was proposed by Charles Loop in 1987 as an improvement over existing subdivision algorithms. The basic idea is to iteratively subdivide each polygon in the mesh into smaller ones, and then adjust the positions of the newly created vertices based on a weighted average of their neighboring vertices. This process is repeated several times, resulting in a progressively smoother surface.

One of the advantages of loop subdivision is that it preserves the overall shape and also has the desirable property of converging to a limit surface that is a piecewise smooth function with continuous first and second derivatives. In 1998, Jos Stam proposed a analytical evaluation method of the Loop subdivision [29] to determine the limit surface. This analytical and differentiable evaluation makes the subdivision scheme ideal for shape optimization as it allows the shape to be smoothed without any additional vertices being introduced and also allows for backpropogations and gradient flows across the subdivision.

2.5 Diffusion Models

Diffusion probabilistic models, or diffusion models for short, have their roots in nonequilibrium thermodynamics in physics [30]. They consider the generative process of a diffusion model as a physical system that is driven away from equilibrium by a random force. The purpose of the model is to learn the dynamics of this system and generate new samples. These models have been applied in deep unsupervised learning to create new latent variable generative models that can learn more complex and realistic data distributions, especially in image synthesis.

Denoising Diffusion Probabilistic Models (DDPM) [31] for image synthesis, which

are a variant of Diffusion Probabilistic Models (DPMs) specifically designed to handle noisy data. DPMs simulate a random walk through a high-dimensional space, where each step corresponds to a change in the generated image. DDPMs introduce a denoising mechanism to the generative process by adding a denoising step to the DPMs.

The idea behind DDPMs is to remove noise from input data while retaining important features by training a denoising function to map the noisy input data to a clean version of the data. The denoising function is trained to minimize the difference between the noisy input data and the clean output data. DDPMs consist of two main steps: a forward process (diffusion process) and a reverse process (denoising step).

What sets diffusion models apart is the forward process $q(x_i)$, which adds Gaussian noise to the data using a fixed Markov chain based on a schedule of variances $(\beta_1, \beta_2, ..., \beta_T)$. DDPM simplifies this process by setting all variances to a constant value, although they can be learned through reparameterization. This means that the forward process of DDPM does not have any learnable parameters.

$$q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}), \qquad q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t\mathbf{I}) \tag{1}$$

The forward process of diffusion models has an important feature: sampling x_t at any timestep t can be accomplished using a closed-form solution without requiring multiplication each time. More information can be found in the DDPM paper. By gradually adding Gaussian noise to the original data x0, we end up with a normal distribution $p(x_T) = \mathcal{N}(x_T; 0, \mathbf{I})$ at timestep T. The idea is that if we can train a neural network to gradually remove noise from generated noisy data, we can start from any randomly generated normal distribution and generate new samples. This is exactly what the reverse process aims to achieve. In the reverse process $p_\theta(x_i)$, our goal is to gradually recover the original data from the noisy images by performing multiple small denoising steps $p_\theta(x_{t-1}|x_t)$, rather than attempting to recover the original data directly. The reverse process can also be defined formally as a Markov chain with learned Gaussian transitions initiated from a random normal distribution $p(x_T)$.

$$p_\theta(x_{0:T}) = p(x_T) \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t), \qquad p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \tag{2}$$

The main objective of DDPM is to acquire knowledge of the reverse process.

Specifically, it is essential to model $\mu_\theta(x_t, t)$ and $\Sigma_\theta(x_t, t)$, as these are the only learnable parameters in $p_\theta(x_{t-1}|x_t)$. To achieve this, they do not teach the covariance matrix $\Sigma_\theta(x_t, t)$ and establish it as $\Sigma_\theta(x_t, t) = \sigma_t^2\mathbf{I}$, where σ_t^2 t is a function of $\beta_{0:t}$, indicating that it is a parameter that depends on time.

For the mean $\mu_\theta(x_t, t)$, DDPM models the forward process posterior means through a reparameterization. Specifically, they optimize a neural network $f_\phi(x)$ with the following objective:

$$L = ||\epsilon_t - f_\phi(x_t)||^2 \tag{3}$$

Where L is the loss to be optimized and ϵ_t is the noise added at timestep t. $f_\phi(x_t)$ is the predicted noise, DDPM uses a U-Net neural network for this. After training the model is able to predict the noise added at a given timestep. To generate novel images we start with randomly initialized gaussian noise and over a set of timesteps we ask the model to predict the noise added. By subtracting this noise over time we can generate images.

This sort of generation is unconditional as it generates images that are just part of the training distribution. However, much research has focused on conditioning the generation on some form of class category [32] or textual prompts [3] by adding cross attention layers in the neural network defined by $f_\phi(x_t)$ changing it to $f_\phi(x_t, y)$ where y would be a representation for the text. The cross attention layers use information n the class/text variable y during training and so at inference time the generation process can be controlled by setting them as desired.

-

Chapter 3

Related Works

3.1 Text to 2D

Image generation is a topic of research that has been studied as much as image recognition and classification. Prior to the advent of deep learning, generation techniques were primarily based on rule-based or statistical methods [33, 34]. These methods involved defining a set of rules or statistical models that could generate images based on certain input parameters or features. However, these approaches were limited in their ability to generate realistic images and often produced low-resolution or blurry results.

With the emergence of deep learning, image generation has been revolutionized. Generative adversarial networks (GANs) [35] have been particularly successful in generating high-quality images that are visually indistinguishable from real images. Further research worked on controlling the generation process by allowing the models to be conditioned on input classes [36], pushing quality of generation while allowing for more controllable generation parameters [37] and introducing novel architectures such as image codebooks to allow for high resolution GAN image generation through VQ-GAN [38]. However, even with these advancements, research in training GANs to be conditioned on text or generating from text was not available apart from small scale works that focused on single class of data like cars or chairs.

Multimodal deep learning discussed in Chapter 2 has paved the way for exciting research in text-driven 2D image generation. One of the key techniques used is the previously discussed CLIP model, which learns a joint embedding space for images and text. This allows researchers to manipulate and generate images using text prompts as guidance. One notable application of this technique is StyleCLIP [39], which uses a distance loss function in the shared embedding space to generate images

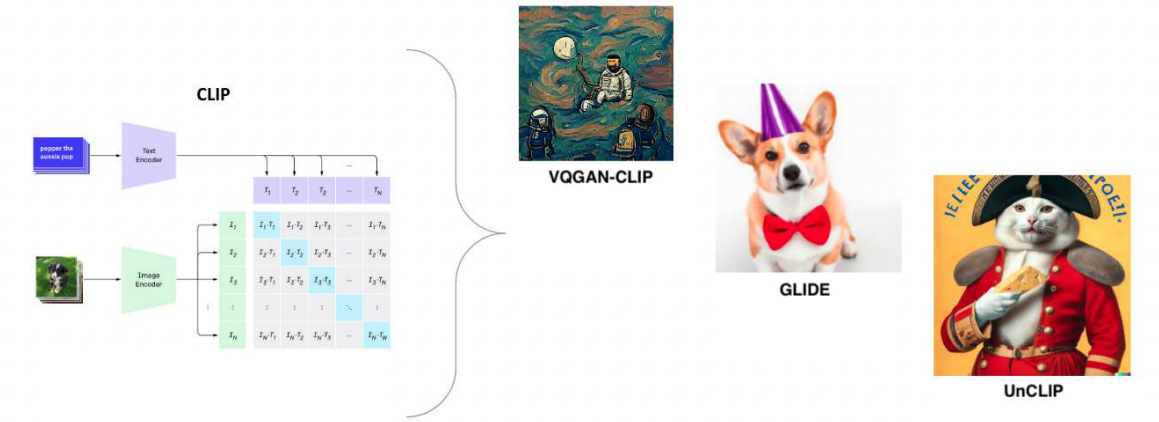

Figure 1: Left: The architecture of CLIP. Right: Generation results from VQGAN-CLIP [1], GLIDE [2] and UnCLIP [3]

that match a given text prompt, this is done by guiding a pretrained GAN model through its learned latent space. Another recent work, VQGAN-CLIP [1], combines the VQGAN image generator with the CLIP model to create high-quality images based on textual prompts. It uses a similar approach to StyleCLIP but instead guides the codebook of the VQGAN rather than latent space embedddings. GLIDE [2] trains a large scale diffusion model to generate high quality images and then guides the diffusion process using CLIP to generate images that match text prompts. Figure 1 shows the CLIP model architecture on the left along with images generated from the techniques mentioned, we see that the quality and expressively of generation has improved dramatically over a short span of time.

The current state of the art in text to image synthesis called UnCLIP [3] has moved away from using CLIP as a guiding factor but it instead uses the embeddings from the CLIP image and text encoder as training data.

3.2 Text to 3D

In contrast to text to 2D, research in text to 3D is an underdeveloped field. However, there are still some notable works that have attempted to allow for custom shape generation. These can be broke up to two main approaches. The first is a generative based approach where a model is trained and developed for the purpose of outputting a 3D shape given some text prompt. The other is an optimization based approach where an initial guess of the shape is iteratively refined to produce the desired shape

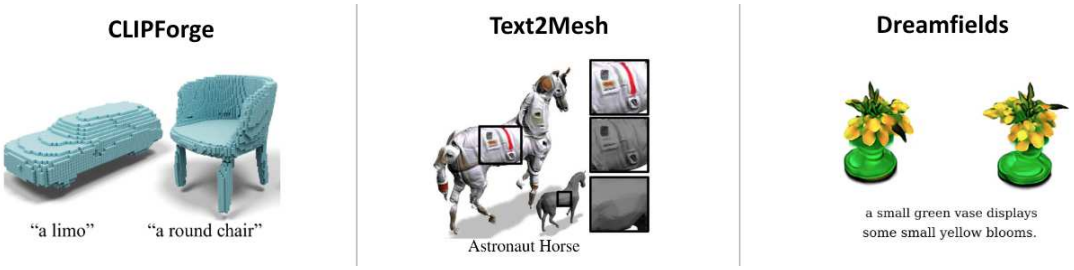

Figure 2: Generation results from CLIP-Forge [4], Text2Mesh [5] and Dreamfields [6]

requested by a text prompt.

Multiple papers [40, 41] proposed a joint embedding between 3D shapes and text, combined with GANs to produce novel outputs. However, these approaches are limited by the lack of available matched 3D models and text descriptions which were restricted by limited shape variation. CLIP-Forge [4] addresses the problem of paired text and 3D models by training an encoder and decoder only on 3D models, and then guiding the decoder with CLIP to produce results that match a text prompt. However, this approach is restricted by the 3D data categories available to train it and does not produce meshes or textures as seen in Figure 2

Some research has focused purely on stylization of exisiting 3D meshes with text prompts. AvatarCLIP [42] and CLIPMatrix [43] are the prominent works in this domain and each focus on deforming a human model slightly and updating its corresponding texture maps. Text2Mesh [5] generalizes this work further by allowing stylization of any mesh through training of a neural network that encodes vertex offsets and vertex colors. As seen in Figure 2 it is able to stylize a preexisting horse mesh but generate the shape on its own.

The closest research to our is a novel text to 3D method called Dreamfields [6] which proposed a zero-shot text-guided generation using a neural radiance field model that is iteratively optimized using CLIP as a loss. It requires raycasting and training a set of neural network parameters, resulting in a large computation overhead even for low-quality generation. Unlike this approach, our figures are all generated on a single 16GB GPU, and the shape, texture, and normal can be individually modified, allowing for unique application scenarios. Figure 2

Chapter 4

Method

4.1 Overview

An overview of our method is shown in Figure 3. We represent a 3D model using three components: (1) a 3D mesh whose vertices $\boldsymbol{V}_0 \in \mathbb{R}^{n\times 3}$ are the control vertices of a Loop [28] subdivision surface $\boldsymbol{V} = S(\boldsymbol{V}_0)$, (2) a texture map T and (3) a normal map $\tilde{\boldsymbol{T}}$. This is a standard way to represent geometric assets in video games and modeling applications. Furthermore, using a texture map allows to decouple the appearance from the geometry and the combination of normal map and subdivision surface control allows us to reduce the number of optimization parameters of the geometry while maintaining rendering details. Our method creates a 3D model by

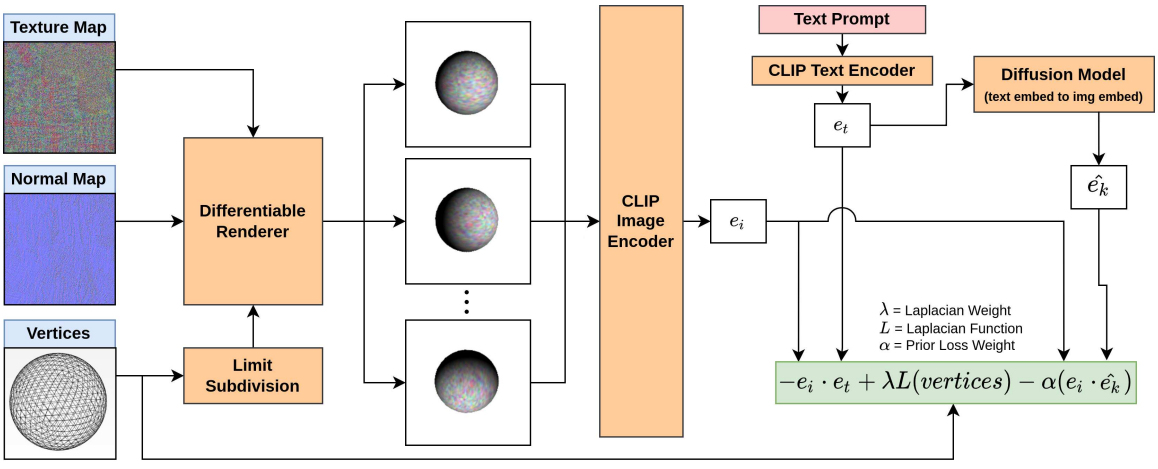

Figure 3: Overview of our optimization pipeline. The differentiable renderer creates views which are encoded and compared to the text encoding as well as the generated image embedding. We optimize for the texture, normal, vertices position.

optimizing these three components using a differentiable renderer. Our rendering pipeline uses the initial control mesh to compute the limit surface V of the Loop subdivision scheme [29]. This limit surface can be computed analytically and it is a differentiable function. The loop subdivision surface V is also, by construction, smooth. Therefore, this surface definition acts as an implicit regularizer and helps avoid triangle inversion during the optimization phase. We render this mesh using a differentiable renderer R [44] from several camera positions $D(\varphi, \theta)$.

We uniformly sample a camera azimuth angle φ from a range of 0° to 360° and for elevation θ we sample from a Beta distribution with $\alpha_d = 1.0$ and $\beta_d = 5.0$ within a range of 0° to 100° this allows the generation to focus on making the object consistent from a single elevation angle giving it a "front view" but the distribution allows other elevations so that textures get painted in for triangles in those regions but the shape does not deform significantly. Using these camera positions and orientation we render a set of images I:

$$\boldsymbol{I} = R(D(\varphi_i, \theta_i), \boldsymbol{V}, \boldsymbol{T}, \tilde{\boldsymbol{T}})$$

Images I_i are encoded using the CLIP image encoder C^I:

$$\boldsymbol{E} = C^I(\boldsymbol{I})$$

Where E represents a set of encodings for each image in I. The input to our method is a text prompt $\boldsymbol{p}$ that is encoded using the CLIP text encoder C^T:

$$\boldsymbol{e}_t = C^T(\boldsymbol{p})$$

As the rendered images as well as the text prompt are now encoded in the same space we can compute the similarity:

$$L_{CLIP}(\boldsymbol{V}, \boldsymbol{T}, \tilde{\boldsymbol{T}}, \boldsymbol{p}) = -\frac{1}{K} \sum_{\boldsymbol{e}_i \in \boldsymbol{E}} \boldsymbol{e}_i^T \boldsymbol{e}_t \tag{4}$$

Note that the encoder functions, C^T and C^I, include a normalization at the end thus these are cosine similarities. As computing the limit loop subdivision surface is differentiable [29] and the renderer is differentiable, our entire pipeline is differentiable using the chain rule.

4.2 Laplacian Regularizer

The uniformly-weighted Laplacian operator, also known as the graph Laplacian, is a mathematical tool used in graph theory to study the properties of graphs. Given an undirected graph, the Laplacian operator is a matrix that encodes the relationships between the vertices of the graph. The matrix is constructed in such a way that the diagonal entries represent the degree of each vertex, while the off-diagonal entries represent the weights of the edges connecting the vertices. The uniformly-weighted Laplacian operator is obtained by setting all the edge weights to 1.

One of the key properties of the uniformly-weighted Laplacian operator is that it is positive semi-definite. This means that all of its eigenvalues are non-negative, which has important implications for the study of graph spectra. The eigenvalues of the Laplacian operator are closely related to the connectivity and structure of the underlying graph, and can be used to extract useful information about the graph. The Laplacian operator is used to define the Laplacian energy of a graph, which is a measure of the "smoothness" of the graph's structure and is used in image processing and computer vision.

We use a laplacian regularizer on the shape of the mesh to maintain the geometry and keep it intact as used in other related work [45]. We use the uniformly-weighted Laplacian operator: $\delta_i = v_i - \frac{1}{|N_i|}\sum_{j \in N_i} v_j$ where N_i is the set of one-ring neighbours for vertex v_i. With this formulation the laplacian regularizer can be given by:

$$L_\delta = \frac{1}{N}\sum_{i=1}^{N}\|\delta_i\|^2 \tag{5}$$

where N is the number of vertices. This minimizes the difference in position between each vertex and the average position of its neighbouring vertices.

4.3 Diffusion Prior

So far as part of our methodology we compare text and image embeddings produced by CLIP. This comparison relies on the assumption that the latent space encoding both is well behaved, in the sense that similar image and text prompts are nearby. This assumption helps produce basic results but research [7] has shown that there exists a "modality gap". In CLIP, the visual and textual inputs are processed separately by a convolutional neural network (CNN) and a transformer-based language model, respectively. While the CNN is trained to extract visual features from images, the

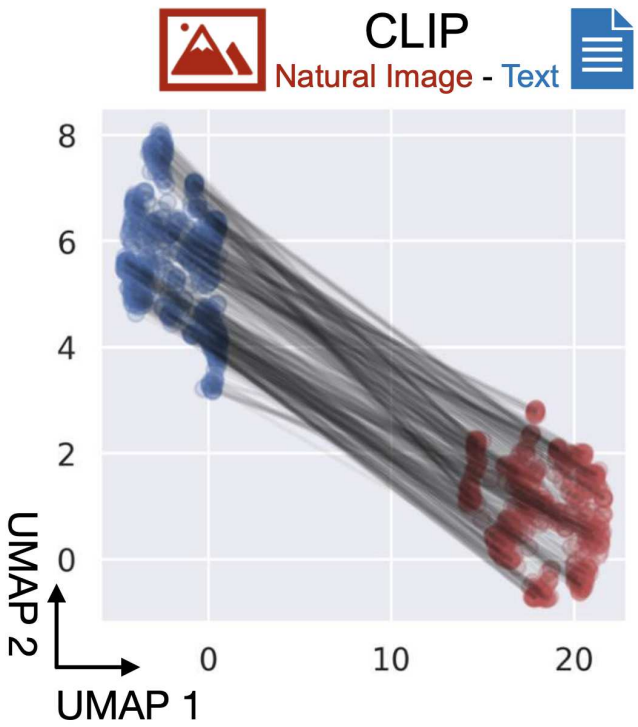

Figure 4: PCA analysis of text and image embeddings showing the modality gap [7]

language model is trained to extract semantic features from text. This creates a gap between the two modalities, as the visual features and semantic features may not have a common representation space. Consequently, when CLIP is used to match images and text, it may not be able to capture the subtle nuances and context-dependent relationships between the two modalities. Figure 4 shows a PCA analysis of CLIP image and text embeddings showing this modality gap.

To further improve results we also train and incorporate a diffusion prior which attempts to generate image embeddings following $p(e_i|e_t)$. We use this to sample image embeddings given a text encoding. Our formulation follows that of [3] and [46]. Once trained, the diffusion sampling process takes input of noise and the CLIP text embedding $\boldsymbol{e}_t$ and after applying the forward process for N time steps the output is a CLIP image embedding which follows $p(e_i|e_t)$.

We pretrain this prior on a 400 million image and text pair dataset [47] so it can sample a relevant CLIP image embedding when given a CLIP text embedding and during optimization time we sample from it using the previously obtained text embedding $\boldsymbol{e}_t$ to get a relevant CLIP image embedding $\hat{\boldsymbol{e}}_{\boldsymbol{k}}$. As the rendered images are encoded in the same space as the output embedding we can also compute a similarity

between them to use as a loss. This reduces the impact of the modality gap and improves generation quality.

$$L_{PRIOR}(\boldsymbol{V}, \boldsymbol{T}, \tilde{\boldsymbol{T}}, \boldsymbol{p}) = -\frac{1}{K} \sum_{\boldsymbol{e}_i \in E} \boldsymbol{e}_i^T \hat{\boldsymbol{e}}_k \tag{6}$$

Since it is conditioned on the text embedding we can use L_{PRIOR} without L_{CLIP}. Practically, in our preliminary experiments we found that combining these losses can be beneficial.

We thus formulate our final problem as an optimization problem with the following objective function:

$$\min_{\boldsymbol{V}_0, \boldsymbol{T}, \tilde{\boldsymbol{T}}} L_{CLIP}(S(\boldsymbol{V}_0), \boldsymbol{T}, \tilde{\boldsymbol{T}}, \boldsymbol{p}) + \lambda_t L_\delta(\boldsymbol{V}) + \alpha L_{PRIOR}(S(\boldsymbol{V}_0), \boldsymbol{T}, \tilde{\boldsymbol{T}}, \boldsymbol{p}) \tag{7}$$

4.4 Implementation Details

Our initial shape is a sphere with 600 vertices. The texture map is initialized with random values and is set to a resolution of 512x512. The normal map has the same resolution but is initialized as a uniform blue image representing uniform normals in +z direction. The Adam [48] optimizer is used for the vertices and texture maps with a decaying learning starting at 0.001 and a batch size of 25. The diffusion prior follows the same configuration setup as [3] except ours is scaled down with reduced network size.

The approach for the laplacian regularization follows that of [45], where the weight, λ, is decayed throughout the optimization process as the shape stabilizes its final form. Initially it is set to a high value when the learning rate is high and then slowly reduces to a minimum value. More specifically, for an epoch t it is defined as $\lambda_t = (\lambda_{t-1} - \lambda_{min}) \cdot 10^{-kt} + \lambda_{min}$. The initial weight and decay parameters are hyperparameters that can be tuned.

The look-at and up vectors of the cameras are set towards the origin and the y-axis respectively. Due to the known texture bias of visual recognition models such as CLIP [49] naively performing the optimization can lead to over emphasis on the texture versus shape. To deal with this we add in some randomization to the view generation process by randomly selecting a camera field of view between 30° to 60° and varying the distance of the camera from the object to between 3.0 to 7.0. This variance in the field of view and distance has a zoom in/out effect that encourages changes in the vertex positions versus only changes in the texture. CLIP takes 224x224 input

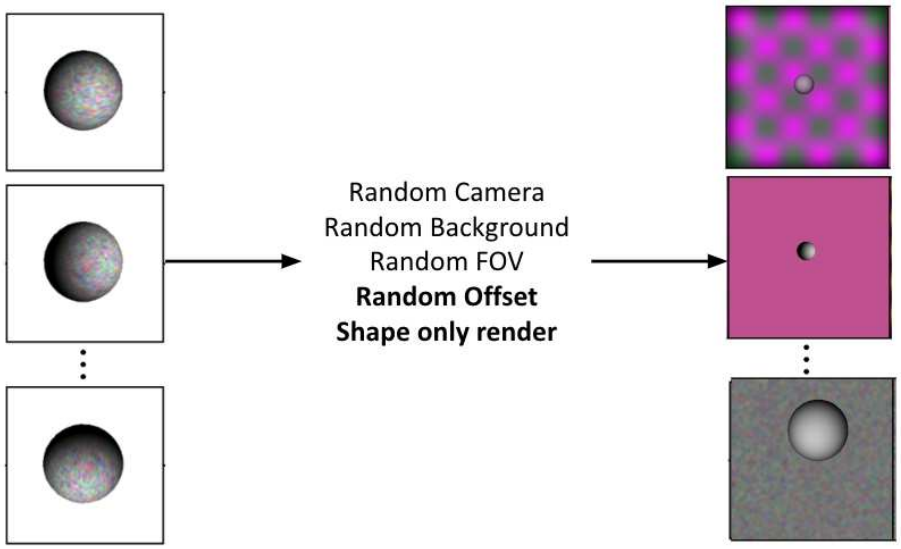

Figure 5: Example of our render augmentations applied to outputs from the differentiable renderer

images but we find that rendering at a larger 512x512 resolution and down scaling to 224x224 improves results, it also plays well with the differentiable render we use [44] since it relies on anti aliasing for gradients and rendering at a larger resolution means more pixels are affected by anti aliasing which reduces gradient noise.

4.5 Augmentations

To improve quality of results we also employ a set of augmentations that help with training:

- Distance of the camera from the object is selected from a uniform distribution between an range of 5.0 to 7.0 units. During training this distance range slowly gets decreased to a range of 3.0 to 5.0 units. This encourages the shape to grow outward initially and prevents CLIP from drawing the input prompt in to the texture right away without changing the shape. The distance ranges and decreasing can be adjusted by the user but the values above are what have worked best for us and all figures use these values.
- The FOV of the perspective camera is randomly augmented to a value between 30° to 60°. Doing this changes the size of the object in the rendering and prevents CLIP from falling in to minimas where it relies overly on the texture

and just paints it in to shape rather that performing any deformations. Since the object is viewed from a distance it also encourages CLIP to grow the shape and mould the object.

- Random background from either solid color, Gaussian noise or checkerboard pattern. We find that this qualitatively improves vibrancy of textures and prevents CLIP from approaching suboptimal solutions where it uses the color of a fixed background towards loss optimization rather than adjusting the texture.
- We also offset the shape to ensure its not always rendered at the center of the image, this is because images that CLIP was trained on are varied and do not always have the object perfectly placed at the center.

In Figure 5 we show how these augmentations affect the renderings, the renders on the left are produced directly from the differentiable render and after augmentationg we get the images on the right. These images are passed to CLIP for encoding.

Chapter 5

Results

We evaluated our methods on a wide variety of prompts and a few different generation scenarios. We first look at the single object generation scenario and compare our method with Dreamfields [6]. We then follow up with additional modeling scenarios unique to our method. Finally we provide quantitative evaluations of our results as well as ablation studies to illustrate the improvement provided by each step of our method.

Figure 6: A 3D scene composed of objects generated using only text prompts: *lamp shade, round brown table, photograph of a bust of homer, vase with pink flowers, blue sofa, pink pillow, painting in a frame, brown table, apple, banana, muffin, loaf of bread, coffee, burger, fruit basket, coca cola can, red chair, computer monitor, photo of marios cap, playstation one controller, blue pen, excalibur sword, matte painting of a bonsai tree; trending on artstation.* (The 3D positioning in the scene was done by a user)

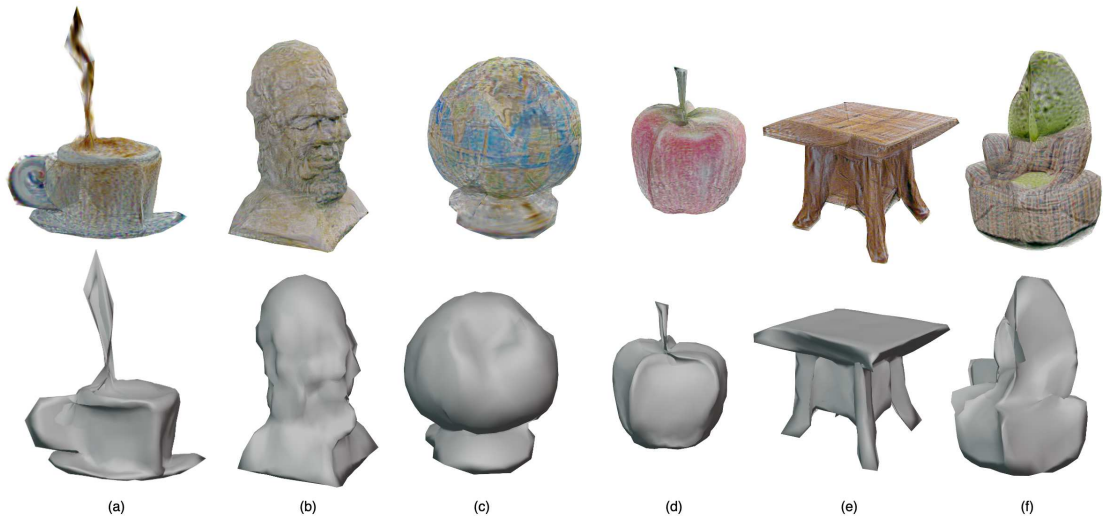

Figure 7: Results from a wide variety of prompts. Top: rendered result. Bottom: 3D mesh. a) "a coffee" b) "a photograph of a bust of homer" c) "Globe" d) "a apple" e) "a brown table" f) "an armchair in the shape of an avocado"

5.1 Single Object Generation

In Figure 6 we illustrate a number of household objects generated using the proposed method. The flexibility of the assets created is illustrated as we import and place them into a 3D scene. In Figure 7 we further illustrate a diverse set of objects and their corresponding shape (removing the texture). Finally in Figure 8 we further show the diversity of possible objects that can be generated using the knowledge of the CLIP model by producing famous landmarks which are visually recognizable. In all these figures we use the CLIP ViT/B-32 model for training.

We also provided visual comparisons to [6]. Fig. 9 shows the results of our methods results with five prompts from [6] with the results shared in their paper and project website. We render the meshes from similar angles. Fig. 9 shows a second comparison with [6] where we chose new prompts and generated the results using the code available online. Note that because their work uses a NeRF representation and requires ray casting it comes with a large resource constraint. Therefore we use the smallest CLIP ViT-B/16 model for the generations and use the medium quality configuration provided in their codebase.

In terms of speed our method is much faster than Dreamfields [6] where each shape took over 24 hours to generate using 4 NVIDIA A100 GPUs. For similar configurations our experiments revealed that our method is faster by a factor of 100

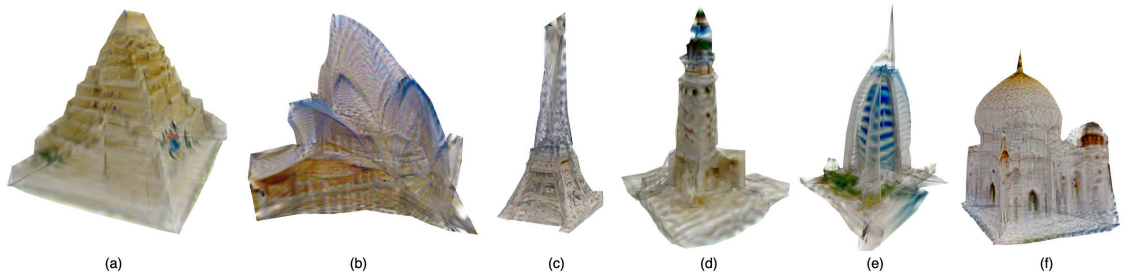

Figure 8: Reconstruction of famous landmark around the world: a) "pyramid of giza " b) "Sydney opera house" c) "Eiffel Tower" d) "lighthouse of alexandria" e) "Burj Al Arab" f) "Taj Mahal"

as each of our shapes required 50 minutes on a single NVIDIA P100 (16GB) GPU. In short the reason for this is two-fold: 1) the number of optimizing parameters in Dreamfields is much higher (all the weights of a complex neural network as opposed to vertex positions, texture and normal maps) 2) our rasterization based rendering is much faster.

5.2 Complex Modeling Scenarios

5.3 Multi Object Generation

Another powerful feature of our method (and unique among NERF based approaches such as [6]) is the flexibility of our optimization framework. The texture and shape are decoupled allowing us to selectively optimize them if needed, and to generate multiple objects in context. This provides a number of unique possibilities for user control of the generation. Additionally since we use meshes it is trivial to combine multiple meshes in to a single mesh while also freezing some vertices and allowing others to be optimized. All this allows us to perform simultaneous optimization of multiple objects as well as separate the shape and texture optimization. This can be be very useful when modeling a scene where some objects have fixed shape while other objects are allowed to vary.

Figure 10 shows an example of this multiple object optimization. In Figure 10a) the text caption used was "boat and red lighthouse", the initial setup was a plane with fixed water texture and 2 spheres on either ends. Vertices and texture for the water were frozen but spheres allowed to optimize. The final result created two distinct shapes for each object in the caption that fits the scene. In Figure 10b) a similar

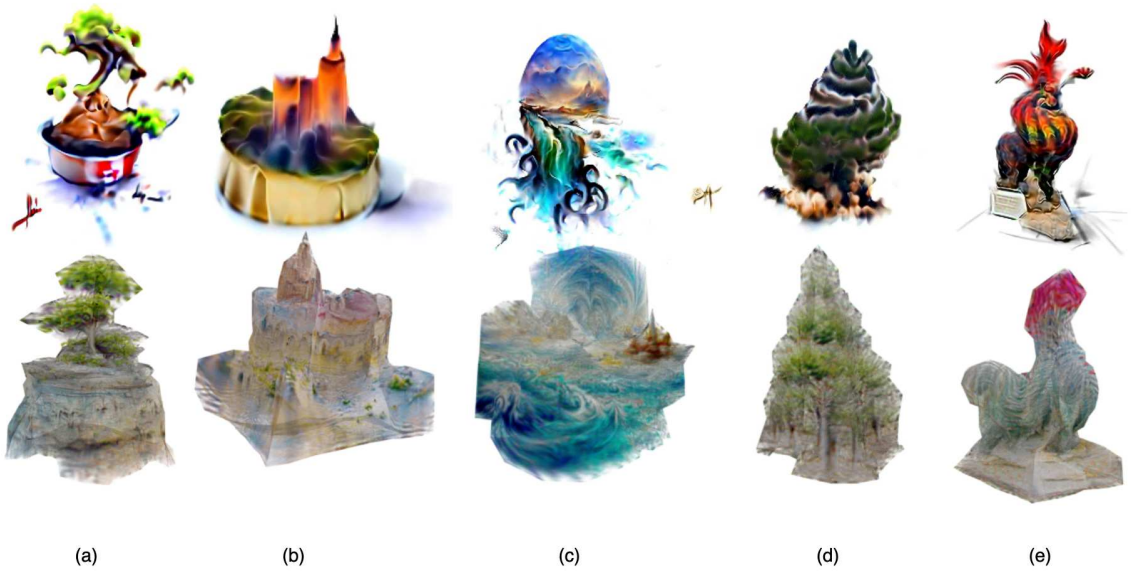

Figure 9: Comparison with [6] results from their paper/project website. Top: results from [6]. Bottom: our results. Prompts: a) "matte painting of a bonsai tree; trending on art station" b) "matte painting of a castle made of cheesecake surrounded by a moat made of ice cream; trending on artstation; unreal engine" c) "a cluster of pine trees are in a barren area" d) "a cluster of pine trees are in a barren area" e) "a sculpture of a rooster"

setup is followed where the carpet and table are static, but the chair and computer monitor are automatically generated from initial spheres. Note that while the starting position of one of the sphere was on the table, we did not specify anywhere explicitly that the monitor should be on the top of the table or that the chair must face the monitor, all of this was inferred implicitly by the model. Figure 11 shows another example of our methods diversity and simultaneous optimization where the sphere allows for shape, texture and normal map optimization while the plane allows only for texture and normal map optimization. We show results for various distinct captions and also note that the texture and normal map of the plane optimize to support the object such as a picnic mat texture appearing when the caption is a "fruit basket on grass"

5.4 Texturing

Although our research is focused purely on generation of 3D shapes and textures, we find that by freezing the shape and allowing textures to be optimized our framework

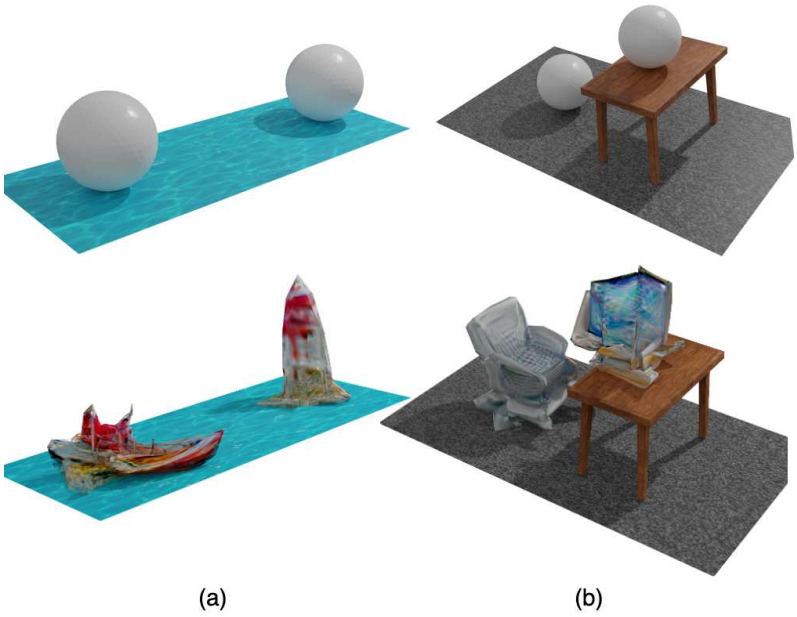

Figure 10: Multiple object optimization. Prompts: a) "boat and red lighthouse" b) "office chair and a desk and a computer monitor"

Figure 11: Multiple object optimization where one of the objects has fixed shape. a) initial shapes. The following are results of the following captions: b) "cactus and sand" c) "wooden boat and blue water" d) "brown wooden table and iranian carpet" e) "fruit basket on grass"

Figure 12: Texture only optimization of a Cow mesh showing how its texture maps change over iteration steps to match the text prompt "a cow"

can be used as a texture generator. This can be further adjusted to generate texture, normal map or both. In Figure 12 we see an example of this where a single cow mesh is kept static but its texture maps are allowed to be optimized to match the prompt "a cow". Similarly in Figure 11 the plane on the floor is optimized in the same way with its shape intact but normal and texture map optimized

Chapter 6

Evaluation

6.1 Quantative Comparision

We quantitatively evaluate our method, comparing it directly with the current closest work of [6]. We follow the same experiment setup outlined in their paper: two shapes are generated per caption for a set of 153 text captions, for a total of 306 generated shapes. During evaluation they are rendered from a held-out pose not seen during training, a CLIP-R precision score [50] is then computed between the held-out pose renderings and the captions used to generate the shapes. The captions used are from [6] and the held out pose is also the same as theirs at a 45° elevation where as training is limited to a 30° elevations, we experiment with different sized CLIP models for generating the shapes and computing the precision.

Table 1 shows the quantitative results of the evaluation. Note that in this evaluation we do not include the diffusion prior loss as the dataset for training the prior contains only CLIP ViT-B/32 embeddings, so we are unable to train a prior that supports the generation model of CLIP ViT-B/16. Regardless, we find that our work outperforms [6] across the generation and evaluation models without it.

Table 1: Quantative comparision of our work with dreamfields on COCO caption object generation

Generation Model	CLIP ViT-B/16		CLIP ViT-B/32	
Evaluation Model	ViT-B/16	ViT-B/32	ViT-B/16	ViT-B/32
Dreamfields [6]	93.5	59.8	74.2	86.6
CLIP-Mesh [Ours]	**96.7**	**67.8**	**75.8**	**91.4**

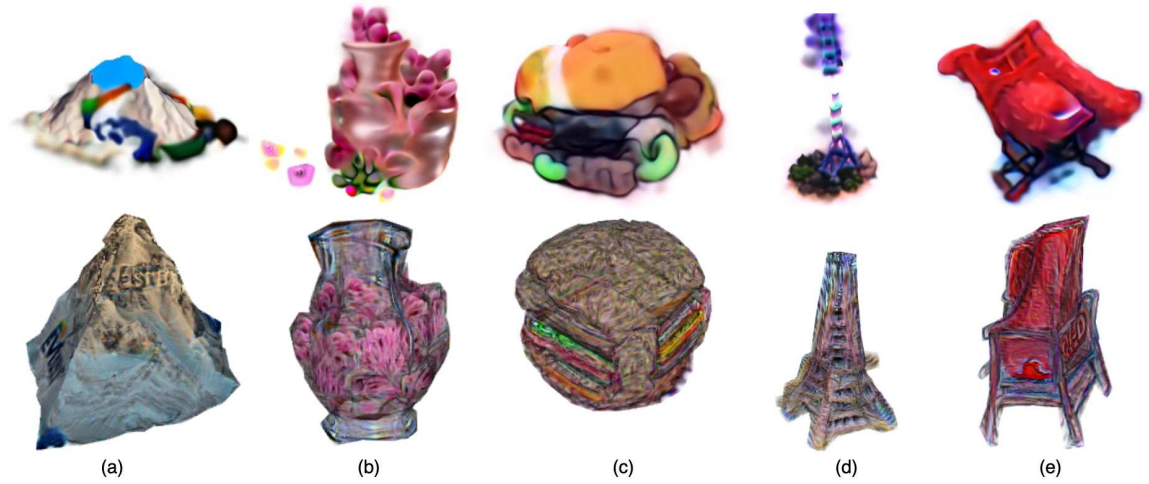

Figure 13: Comparison with [6]. Shapes generated using CLIP ViT/B-16 Top: results from [6]. Bottom: our results. Prompts: a) "mount everest" b) "a vase with pink flowers" c) "a hamburger" d) "Eiffel tower" e) "a red chair"

6.2 Ablation Study

In Table 2 an ablation study is shown for the various components of our pipeline. We start from a stripped down version of our method (baseline) and sequentially add in the components of the method. We follow the same evaluation methodology as in Table 1 but a single shape is generated per caption here instead of two as we found that it does not have a significant impact on the metric and reduces the time required per evaluation. Our results show that the limit subdivision provides an improvement across all retrieval models. We then add the image augmentations which both provide improvements, offsetting the mesh from the center of the image provides the largest boost to the final results. Similarly, rendering the images at a higher resolution and then linearly scaling to the CLIP 224x224 resolution does improve results in all cases except for the largest ViT-L/14 model where it hurts performance. We get our best overall results when adding the prior loss.

Our experiments also show that the limit subdivision improves the results both quantitatively and qualitatively. Table 2 shows the quantitative improvement and in Fig. 14 we show meshes for the same text prompt with and without limit subdivision. Left is limit subdivision and right is without it - we find that it reduces mesh tangling and leads to better triangulation.

Table 2: Ablation study on the R-Precision quantitative metric where higher score is better. We observe that starting from a baseline approach, adding limit subdivision, augmentation, large rendering, and the generative prior systematically improves performance

Method (CLIP B/16)		CLIP R-Precision ↑		
		B/16	*B/32*	*L/14*
Shape	Baseline Method	75.8	41.8	50.9
	+ Limit Subdivision	77.7	47.7	53.5
Augmentations	+ Background	81	47.7	58.8
	+ Repositon Shape	90.1	60.5	73.2
Render	+ 512^2 renders	92.1	62.7	70.5
Prior	+ Prior Loss	91.5	77.7	74.5

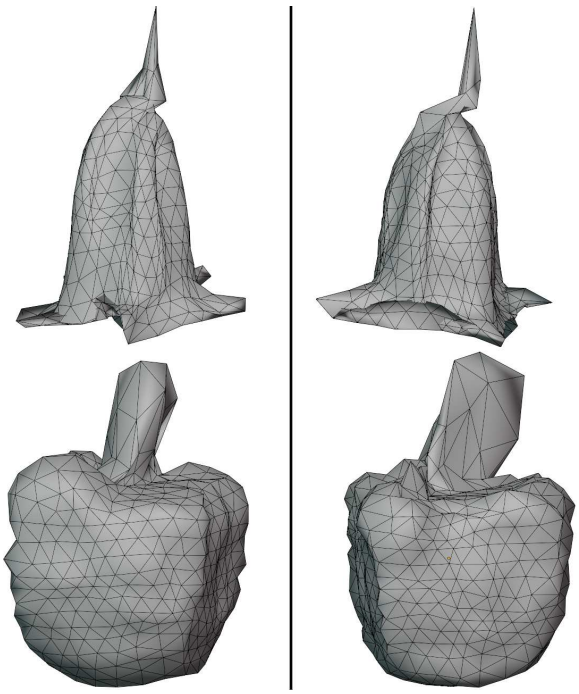

Figure 14: Final output results (left column) with and (right column) without limit subdivision for the text prompts "Burj Al Arab" (top) and "A fruit basket" (bottom)

Chapter 7

Conclusions

7.1 Contributions

We have demonstrated a method for generating diverse 3D objects in different modeling scenarios using only an input text prompt. The results consist of a mesh, texture map and normal map which allow them to be directly loaded to be used as assets in games and modelling applications. While the work we propose provides interesting results there are some limitations of our method.

Genus The genus of the generated object is set by the initial template mesh. We address this issue partially by allowing a transparency channel in the texture, but a more principled approach is desirable.

CLIP Limitations Using an image model to generate 3D shapes comes with its own challenges, since the model is trained with images it often projects artifacts to the mesh. Some examples of this can be seen in Figure 8 where the pyramid has small people on its side and Figure 13 where the mount Everest has the text "Everest" on its side and tip, note that we find using the larger CLIP ViT/B-32 model alleviates the text issue.

In future work we will aim to further improve shape based constraints and explore methods to provide more user control in the generative process.

7.2 Future Works

The speed of development in generative deep learning is unprecedented and even after publication of our work there are already multiple papers focused on text

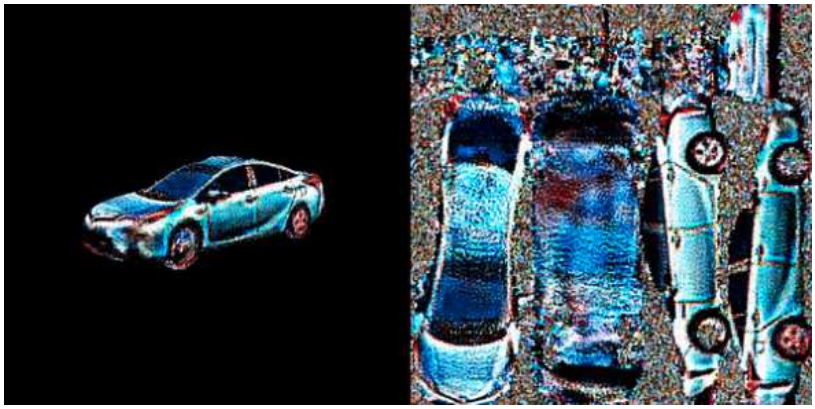

Figure 15: Using SDS loss proposed in Dreamfusion [8] to texture a car model, diffusion based losses outperform CLIP

Figure 16: Left: Depth map input to depth conditioned diffusion models. Right: Three output images generated by the model showing its ability to follow input depth map

to 3D. The current state of the art method called Dreamfusion [8] is a successor to the work Dreamfields and introduces a novel loss that replaces CLIP with pretrained text to image diffusion models. Specifically it uses DDPM based models mentioned previously and relies on their generative capability over CLIP. This improves performance on quantative and qualitative metrics.

We investigated this loss function and found that it is well suited to neural representations like NeRF but struggles with our mesh based framework. However, it can be used in the same texturing framework as discussed in section 5.4 and we find that the quality is much better and more akin to a realistic texture generation such as for the car result shown in Figure 15

Although this new loss function improves qualities it still suffers from certain problems that affect our work as well. The problem of 3D reconstruction based on 2D information is itself very difficult for the same reason that inverse rendering is hard: there exist many 3D shapes that result in similar 2D images. The optimization landscape is very non-convex and therefore we often find bad local minima such as situations where instead of creating a shape the method paints all the requested scene information on to a flat plane. Without any geometric prior injection it will be difficult to overcome these issues

Recent open source text based diffusion models also incorporate conditioning on depth maps, this allows injecting some shape information in to the text to image generation pipeline, in Figure 16. Exploiting the depth consistency of these newer diffusion models to create accurate shapes is an interesting avenue of research that may help in the quest for robust text to 3D shape generation.

References

[1] K. Crowson, S. Biderman, D. Kornis, D. Stander, E. Hallahan, L. Castricato, and E. Raff, "Vqgan-clip: Open domain image generation and editing with natural language guidance," in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXVII*, pp. 88–105, Springer, 2022. vii, 11

[2] A. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. McGrew, I. Sutskever, and M. Chen, "Glide: Towards photorealistic image generation and editing with text-guided diffusion models," *arXiv preprint arXiv:2112.10741*, 2021. vii, 11

[3] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," 2022. vii, 5, 9, 11, 16, 17

[4] A. Sanghi, H. Chu, J. G. Lambourne, Y. Wang, C.-Y. Cheng, M. Fumero, and K. R. Malekshan, "Clip-forge: Towards zero-shot text-to-shape generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18603–18613, 2022. vii, 12

[5] O. Michel, R. Bar-On, R. Liu, S. Benaim, and R. Hanocka, "Text2mesh: Text-driven neural stylization for meshes," *arXiv preprint arXiv:2112.03221*, 2021. vii, 12

[6] A. Jain, B. Mildenhall, J. T. Barron, P. Abbeel, and B. Poole, "Zero-shot text-guided object generation with dream fields," 2022. vii, viii, 12, 20, 21, 22, 23, 26, 27

[7] V. W. Liang, Y. Zhang, Y. Kwon, S. Yeung, and J. Y. Zou, "Mind the gap: Understanding the modality gap in multi-modal contrastive representation learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 17612–17625, 2022. vii, 15, 16

[8] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall, "Dreamfusion: Text-to-3d using 2d diffusion," *arXiv*, 2022. viii, 30

[9] N. Mohammad Khalid, T. Xie, E. Belilovsky, and T. Popa, "Clip-mesh: Generating textured meshes from text using pretrained image-text models," in *SIGGRAPH Asia 2022 Conference Papers*, pp. 1–8, 2022. 3

[10] J. F. Hughes, M. S. McGuire, J. Foley, D. F. Sklar, S. K. Feiner, K. Akeley, A. Van Dam, and J. D. Foley, *Computer graphics*. Boston, MA: Addison-Wesley Educational, 3 ed., Feb. 2009. 4

[11] H. Xie, H. Yao, X. Sun, S. Zhou, and S. Zhang, "Pix2vox: Context-aware 3d reconstruction from single and multi-view images," in *ICCV*, 2019. 4

[12] M. Tucsok, S. H. Gazani, K. Gupta, and H. Najjaran, "3d reconstruction from 2d images: A two-part autoencoder-like tool," *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 538–543, 2022. 4

[13] A. X. Chang, T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "Shapenet: An information-rich 3d model repository," *CoRR*, vol. abs/1512.03012, 2015. 4

[14] J. Reizenstein, R. Shapovalov, P. Henzler, L. Sbordone, P. Labatut, and D. Novotný, "Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction," *CoRR*, vol. abs/2109.00512, 2021. 5

[15] T. Ridnik, E. B. Baruch, A. Noy, and L. Zelnik-Manor, "Imagenet-21k pretraining for the masses," *CoRR*, vol. abs/2104.10972, 2021. 5

[16] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds.), vol. 33, pp. 6840–6851, Curran Associates, Inc., 2020. 5

[17] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal deep learning," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, (Madison, WI, USA), p. 689–696, Omnipress, 2011. 5

[18] X. Hu, Z. Gan, J. Wang, Z. Yang, Z. Liu, Y. Lu, and L. Wang, "Scaling up vision-language pre-training for image captioning," *CoRR*, vol. abs/2111.12233, 2021. 5

[19] K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," *CoRR*, vol. abs/1502.03044, 2015. 5

[20] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," *CoRR*, vol. abs/2103.00020, 2021. 5

[21] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," *CoRR*, vol. abs/2010.11929, 2020. 6

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017. 6

[23] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik, "End-to-end recovery of human shape and pose," *CoRR*, vol. abs/1712.06584, 2017. 6

[24] Y. Jiang, D. Ji, Z. Han, and M. Zwicker, "Sdfdiff: Differentiable rendering of signed distance fields for 3d shape optimization," in *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 6

[25] G. Pavlakos, L. Zhu, X. Zhou, and K. Daniilidis, "Learning to estimate 3d human pose and shape from a single color image," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 459–468, 2018. 6

[26] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *ECCV*, 2020. 7

[27] L. Yariv, J. Gu, Y. Kasten, and Y. Lipman, "Volume rendering of neural implicit surfaces," in *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021. 7

[28] C. Loop, *Smooth Subdivision Surfaces Based on Triangles*. PhD thesis, January 1987. 7, 13

[29] J. Stam, "Evaluation of loop subdivision surfaces," in *SIGGRAPH'98 CDROM Proceedings*, Citeseer, 1998. 7, 14

[30] J. Kurchan, "Fluctuation theorem for stochastic dynamics," *Journal of Physics A: Mathematical and General*, vol. 31, no. 16, p. 3719, 1998. 7

[31] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851, 2020. 7

[32] J. Ho, C. Saharia, W. Chan, D. J. Fleet, M. Norouzi, and T. Salimans, "Cascaded diffusion models for high fidelity image generation.," *J. Mach. Learn. Res.*, vol. 23, no. 47, pp. 1–33, 2022. 9

[33] L.-Y. Wei and M. Levoy, "Fast texture synthesis using tree-structured vector quantization," in *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, (USA), p. 479–488, ACM Press/Addison-Wesley Publishing Co., 2000. 10

[34] S. Dass, A. Jain, and X. Lu, "Face detection and synthesis using markov random field models," in *2002 International Conference on Pattern Recognition*, vol. 4, pp. 201–204 vol.4, 2002. 10

[35] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, and Y. Bengio, "Generative adversarial networks, 1–9," *arXiv preprint arXiv:1406.2661*, 2014. 10

[36] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014. 10

[37] T. Karras, M. Aittala, S. Laine, E. Härkönen, J. Hellsten, J. Lehtinen, and T. Aila, "Alias-free generative adversarial networks," in *Proc. NeurIPS*, 2021. 10

[38] P. Esser, R. Rombach, and B. Ommer, "Taming transformers for high-resolution image synthesis," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12873–12883, 2021. 10

[39] O. Patashnik, Z. Wu, E. Shechtman, D. Cohen-Or, and D. Lischinski, "Styleclip: Text-driven manipulation of stylegan imagery," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2085–2094, October 2021. 10

[40] K. Chen, C. B. Choy, M. Savva, A. X. Chang, T. Funkhouser, and S. Savarese, "Text2shape: Generating shapes from natural language by learning joint embeddings," *arXiv preprint arXiv:1803.08495*, 2018. 12

[41] K. Fukamizu, M. Kondo, and R. Sakamoto, "Generation high resolution 3d model from natural language by generative adversarial network," *arXiv preprint arXiv:1901.07165*, 2019. 12

[42] F. Hong, M. Zhang, L. Pan, Z. Cai, L. Yang, and Z. Liu, "Avatarclip: Zero-shot text-driven generation and animation of 3d avatars," *arXiv preprint arXiv:2205.08535*, 2022. 12

[43] N. Jetchev, "Clipmatrix: Text-controlled creation of 3d textured meshes," *arXiv preprint arXiv:2109.12922*, 2021. 12

[44] S. Laine, J. Hellsten, T. Karras, Y. Seol, J. Lehtinen, and T. Aila, "Modular primitives for high-performance differentiable rendering," *ACM Transactions on Graphics*, vol. 39, no. 6, 2020. 14, 18

[45] J. Hasselgren, J. Munkberg, J. Lehtinen, M. Aittala, and S. Laine, "Appearance-driven automatic 3d model simplification," in *Eurographics Symposium on Rendering*, 2021. 15, 17

[46] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851, 2020. 16

[47] C. Schuhmann, R. Vencu, R. Beaumont, R. Kaczmarczyk, C. Mullis, A. Katta, T. Coombes, J. Jitsev, and A. Komatsuzaki, "Laion-400m: Open dataset of clip-filtered 400 million image-text pairs," *ArXiv*, vol. abs/2111.02114, 2021. 16

[48] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015. 17

[49] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel, "Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness.," in *International Conference on Learning Representations*, 2019. 17

[50] D. H. Park, S. Azadi, X. Liu, T. Darrell, and A. Rohrbach, "Benchmark for compositional text-to-image synthesis," in *NeurIPS Datasets and Benchmarks*, 2021. 26