Operational 3D Mesh Synthesis with Geometric and Diffusion Priors

Diego Martí Monsó

Master's thesis

Operational 3D Mesh Synthesis with Geometric and Diffusion Priors

Diego Martí Monsó

September 04, 2024

Chair of Data Processing
Technische Universität München

Diego Martí Monsó. *Operational 3D Mesh Synthesis with Geometric and Diffusion Priors.* Master's thesis, Technische Universität München, Munich, Germany, 2024.

Supervised by Prof. Dr.-Ing. Klaus Diepold and Supervisor; submitted on September 04, 2024 to the Department of Electrical and Computer Engineering of the Technische Universität München.

Chair of Data Processing, Technische Universität München, 80290 München, Germany, `http://www.ldv.ei.tum.de/`.

Abstract

Recent advancements in 3D generative frameworks that leverage strong 2D diffusion priors for guidance have shown promise in generating high quality 3D assets from text prompts. However, established workflows optimize a neural scene representation and extract a mesh after convergence via a destructive operation. The sampled meshes do not meet the quality requirements for use in production environments because the shape and appearance of the mesh is unrealistic, and the surfaces have irregular topology and UV mappings. Therefore, we propose a novel approach that incorporates a geometric prior in the form of a 3D morphable model, resulting in a class-specific generative model capable of producing production-ready meshes through score distillation sampling. Our method yields meshes with artist-designed topology, skinning weights, and UVs, as well as realistic shapes and colors, making the meshes immediately suitable for practical applications. Furthermore, our approach is interpretable and outperforms existing baselines in terms of text-to-3D mesh alignment.

Contents

1 Introduction

In today's digital era, 3D content is the backbone of innovation across industries such as gaming, film, virtual reality, architecture, and product design, driving both creativity and technological advancement. Despite the undeniable importance of 3D content in these fields, the process of creating high-quality 3D models remains a formidable challenge. Traditional methods of 3D model creation are not only technically demanding but also resource-intensive, often requiring a combination of artistic talent, deep technical knowledge, and extensive time commitments. For instance, creating a detailed 3D model from scratch typically involves stages such as conceptualization, sculpting, topology refinement, texturing, rigging, and rendering. Each part of the process can take weeks or even months to complete. These complexities make 3D content creation largely inaccessible to non-professional users. In addition, there is growing pressure on 3D content creators to deliver sophisticated 3D content quickly and cost-effectively, as audiences and consumers increasingly expect high-quality visuals and interactive experiences. Thus, we identify the need for automated systems capable of generating production-ready 3D content from simple inputs, such as text descriptions or reference images.

The task of automatically generating 3D models has been a topic of computer vision and graphics research for decades [40]. The goal is to learn a predictive model that maps an input text prompt to an output mesh that is *operational* for downstream industrial applications, i.e., the mesh is ready to be used in production environments without additional human intervention. The core challenge lies in ensuring that the generated 3D models satisfy a set of structural requirements concerning predominantly the geometry, topology, texture coordinates, and rigging that are needed for the mesh to be ready for production. For instance, an operational model would not only feature accurate geometry but also have quadrangular topology optimized for deformation, UV mapping that avoids visible texture seams, and skinning weights that allow for realistic pose control. However, these conditions are hard to formalize due to the subjective nature of artistic requirements and the technical diversity of production environments, which complicate the development of universal metrics for production-readiness. Not only artistic considerations such as style or consistency play a role, but also technical factors like hardware or the rendering engine may have great influence. Due to the lack of a formalization of artistic and technical requirements or concrete metrics to optimize for, most research efforts focus instead on improving the perceptual quality of the synthesized 3D assets when rendered as an image [26, 40, 41, 43, 44, 56, 67, 71, 73, 75].

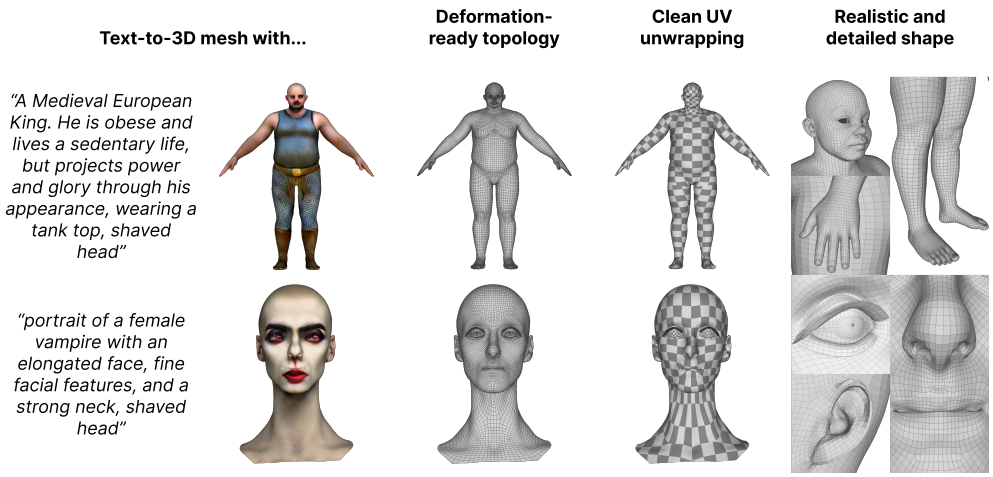


Figure 1.1: We present a class-specific text-to-3D generative model that produces meshes that are operational for industrial application without requiring additional human intervention. At the core of our method lies a 3d morphable model that acts as a geometric prior, which regularizes the topology and the UVs of the mesh, as well as the distribution of shapes. We sample novel meshes via score distillation sampling through a differentiable renderer. For supervision, we leverage a pre-trained multi-view diffusion prior.

These concurrent works predict a 3D model parameterized as a neural scene representation, commonly a Neural Radiance Field (NeRF) [50] or Gaussian Splatting [32]. Then, the surface mesh is extracted from the volume via algorithms based on marching cubes or marching tetrahedra [46, 65, 66, 72]. While neural representations excel at rendering high-quality images differentiably and can potentially capture the spatial structure of a figure up to an interpolation error (and the later discretization error introduced by the mesh extraction step), the sampled shapes suffer from artifacts such as fused mesh components, high genus, the Janus problem, or content drift [44, 67]. Moreover, the topological structure of the mesh – and, by extension, all features attached to the topology, like UVs and rigs – is lost. Therefore, some recent works [77] aim to recover a topological landscape of the mesh graph that is more favorable for production settings by post-processing the extracted mesh with retopology and atlasing algorithms [17]. Nevertheless, the benefit is hard to quantify.

To address the research gap of automatically generating operational 3D meshes, we propose a novel class-specific generative model that allows to program the aforementioned structural properties of the output mesh at design time of the algorithm. We introduce a *geometry prior* that offers a practical guarantee that the synthesized mesh is operational for industrial applications. Additionally, the geometry prior regularizes the shape space to remain close to the manifold of valid shapes. Furthermore, we employ a pre-trained *diffusion prior* for guidance via score distilla-

tion sampling (SDS) [56]. Thus, our generative model can be adapted to new asset classes without retraining. We also provide a comprehensive qualitative definition of the basic surface attributes that 3D meshes must meet in order to be viable for mainstream industrial applications. Finally, we benchmark our approach against state-of-the-art methods, achieving higher quantitative scores on multi-view alignment between the input prompt and the sampled mesh. An overview can be seen in Fig. 1.1.

At its core, our method consists of three differentiable components that allow to sample an operational mesh as a gradient-based optimization of a parametric mesh model. Firstly, we construct the geometric prior as a 3D morphable model (3DMM) [14] from our own curated dataset of mesh blend shapes. We also show that our general framework is in principle compatible with other 3DMMs, such as the ICT FaceKit [39]. Secondly, we build a differentiable renderer as a combination of a differentiable rasterizer [37] for mesh rasterization and Instant-NGP [52] for RGB interpolation. Lastly, we use Stable Diffusion [58] as our pre-trained diffusion prior, which acts as a critic to supervise the optimization process of the mesh via gradient descent.

2 Related Work

The field of 3D generation has seen remarkable advancements over the past decade, driven primarily by the success of generative AI in images and videos [24, 53, 58]. These advancements have led to the development of high-quality and diverse 3D models that are essential for applications in video games, movies, virtual reality, and other immersive experiences. Following [40], the field of 3D generation can be broadly divided into several key areas: (i) the 3D representations that are used; (ii) the generative models, which range from feedforward generation (e.g., GANs [15]), optimization-based generation with CLIP [30, 57] like [25], procedural generation (commonly used for natural environments like terrains or plant structures), to novel view synthesis, which is the backbone of our method; and (iii) the datasets.

In our literature review, we focus on novel view synthesis approaches that use 2D diffusion models. The advantage of using pretrained image diffusion models for 3D generation is that the model can leverage large-scale image datasets such as LAION-5B [63], which consists of over 5 billion image-text pairs. Generative models rely heavily on the availability of large datasets to be trained on, and common 3D datasets such as ShapeNet [9] or Objaverse-XL [13], with around 50 thousand and 10 million assets each, are orders of magnitude smaller than their image counterparts. Besides, there is less stylistic or structural consistency in 3D datasets and there is a lack of labeled data. Models that build on the LRM [26, 73] have recently shown success in image-to-3D synthesis. These models do not take text prompts as input because there is not enough labeled 3D data to train on.

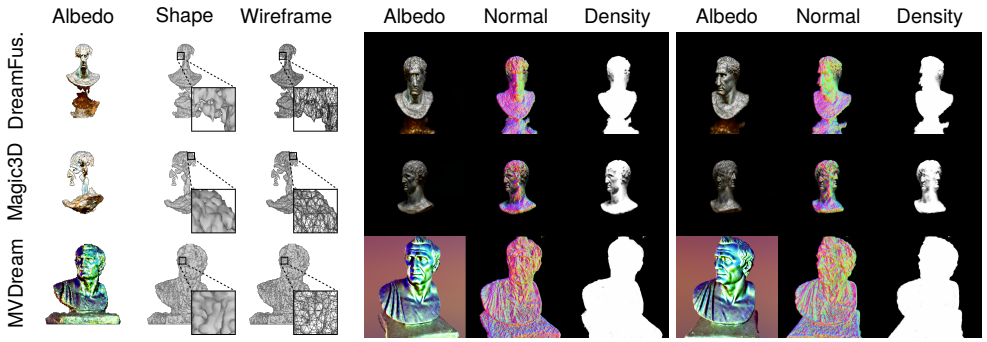
DreamFusion [56] originally demonstrates 3D synthesis from a text prompt via score distillation sampling (SDS) with pretrained diffusion models. The DreamFusion algorithm initializes a NeRF randomly and then samples random views iteratively. The NeRF renderings are noised and fed as input to the SDS loss with the pre-trained Imagen model [60] at 64×64 resolution as the backbone. The neural representation is then optimized via gradient descent. As a last step, a mesh can be extracted from the synthesized NeRF via Marching Cubes [46, 66], Marching Tetrahedra [65], or algorithms that rely on different sets of heuristics like Nerf2Mesh [72]. For more details about SDS, please refer to 3.3.

After DreamFusion, several works have built upon the SDS framework and further explored the SDS objective to propose modifications that aim to improve performance and decrease generation time incrementally. Usually, these changes affect either the scene representation under optimization, the architecture of the diffu-

sion model, or the objective formulation itself. We discuss these approaches in the following.

Modifications to the scene representation under optimization. NeRFs [50] are known to be expensive to sample via volume rendering, and also require many iterations to converge. Therefore, researchers have explored combining SDS with a variety of 3D representations, such as NeRFs and variants like mip-NeRF [7, 16, 30, 43, 50, 56, 75], TriPlanes [26, 73], multi-resolution hash encodings [41, 52, 67], or Gaussian Splatting [32, 71] as their 3D representation to accelerate the synthesis of 3D objects from a text prompt. We highlight for example the case of DreamGaussian [71], which leverages the recent breakthrough of 3D Gaussian Splatting [32] to achieve faster rendering and convergence in less iterations. However, DreamGaussian lacks the visual quality that concurrent methods achieve with NeRF.
Regardless of the choice, no neural 3D representation is aware of the mesh surface geometry (i.e., the topology). We thus choose a 3DMM [14] as our 3D shape representation, as 3DMMs allow to pre-define the topology of the output mesh. For the mesh color, we use the efficient InstantNGP [52], which is also used by other works like [41, 67], allowing for fast iterations when combined with the NVDiffRast rasterizer [37]. We are not the first work to optimize over meshes. Magic3D [41] proposes a two-stage optimization approach, where a mesh is extracted from an optimized InstantNGP in the first step, and iteratively refined via SDS in the second part. Nevertheless, the extracted mesh still has the usual artifacts of isosurface extraction algorithms [46] and the mesh refinement stage does not modify the topology or the genus of the mesh. Similar to us, AvatarCLIP [25] and DreamWaltz [28, 29] also realize that a 3DMM can be used as a shape prior to guide the SDS process to a mode, allowing to converge to more realistic shapes. However, both methods either perform destructive operations on the mesh or use the 3DMM only to regularize the space of a neural field.

Modifications to the architecture of the diffusion model. One recent line of research focuses on adapting the architecture of the diffusion prior to better suit the SDS loss. While SDS is effective at providing optimization guidance from a pre-trained 2D diffusion network to some 3D scene, the SDS objective suffers from information loss at every iteration due to the locality of the rendering process. The diffusion model is trained in image space and can thus only operate on a lossy projection of the 3D scene to the 2D image plane. For instance, when denoising the front view of a scene, the model is unaware of all the occluded parts of the scene (such as the back view). Furthermore, the information loss caused by the projection step also affects the backpropagation step, which only updates the scene locally where components of the scene contribute to the rendering. Therefore, ar-



(a) Rendering of the extracted mesh from a frontal view.

(b) Volume rendering from a frontal perspective.

(c) Volume rendering at a 24º azimuth angle.

Figure 2.1: We sample the prompt "A bust of Julius Caesar" from DreamFusion [56] (first row), Magic3D [41] (second row), and MVDream [67] (third row). In Fig. 2.1a, we show that the extracted meshes have irregular surfaces and topology. We illustrate the underlying volume representation that encodes the 3D asset as a cloud of color in Figs. 2.1b and 2.1c. Here, we observe the Janus problem and content drift when slightly rotating the camera for Magic3D, as well as oversaturated colors for MVDream. Moreover, the uneven distribution of the density is responsible for the unsmooth surfaces of the extracted mesh, along with the holes and floating blobs of material.

tifacts like the Janus problem and content drift become common. In Fig. 2.1, we explore these effects for three baselines: DreamFusion [56], Magic3D [41], and MVDream [67].

Many methods have emerged that aim to finetune a pre-trained diffusion model, with modifications to the architecture of the U-Net [59], to become pose-aware and multi-view consistent. Most notably, MVDream [67] expands the 2D self-attention [3, 49] of StableDiffusion [58] to achieve multi-view diffusion. Similar to Zero-123 [43] and Zero-123-XL [13], the network is trained and multi-view renderings of 3D assets.

Multiview consistent diffusion models greatly improve visual quality of the sampled 3D models, as is evidenced in Fig. 2.1. Still, the synthesized 3D scenes are still not entirely multi-view consistent, which can be attributed mainly to two reasons. Firstly, the multiple views that are denoised together at every iteration do not cover the entire visible surface of the 3D model. For example, SyncDreamer [44] and MVDream independently propose different sampling strategies to cover the surface of the 3D asset by sampling around the azimuth range with different elevations. But any projection of the 3D model to 2D planes is prone to leave certain visible gaps to the diffusion model, unless we sample with a very high density. Secondly, despite the improved consistency across different views, the modified diffusion model itself is still not entirely multiview consistent, leading to artifacts that can accumulate over SDS iterations. With architectural and training improvements driven by video dif-

fusion models [10, 24], recent methods like Cat3D [16] have achieved single-step reconstruction of a 3D representation without the need for SDS. The weights of the model that they train are not publicly available.
Based on these observations, we opt to use the multi-view architecture provided by MVDream [67]. In our case, we circumvent the aforementioned problems caused by the locality of each SDS iteration by optimizing a scene representation that is globally 3D consistent. Specifically, gradient updates to a localized subset of scene components affect the scene parameterization globally in a way that is geometrically consistent according to the shape prior that is defined by a 3DMM [14].

Alternative formulations of the SDS objective. ProlificDreamer [75] introduces the concept of variational score distillation sampling (VSD), which is a particle-based approach where several SDS optimizations are run in parallel in a variational framework. While VSD greatly improves the generation quality, it has not seen widespread adoption due to the high cost of running SDS in a particle-based simulation. Other methods such as Perp-Neg [2], Delta Denoising Score [20], and LMC-SDS [1] provide alternative formulations of the SDS objective that have not either become the standard in SDS research due to increased generation times and limited improvement. Nevertheless, we take inspiration from their analysis to justify some of the choices that we make. Recently, Lukoianov et al. [48] demonstrate that SDS is similar to a high-variance version of Denoising Diffusion Implicit Models (DDIM) [69], which is a popular scheduler to sample from diffusion models in few steps. Thus, the authors derive a new formulation of SDS that shows great promise in improving the perceptual quality of neural fields sampled via SDS, although the algorithm is also considerably more costly than vanilla SDS.

We focus on single verticals of 3D generation. In contrast to previous methods, we do not aim to sample from the entire distribution 3D assets that have been scraped from the internet and put into large datasets like Objaverse-XL [13] and ShapeNet [9], or that can be derived from image datasets like Laion [63]. Instead, we focus on solving the entire vertical of synthesizing high-quality, operational meshes for single asset classes. We use our own proprietary dataset of 3D meshes, which has been curated to meet strict quality and consistency standards, while still leveraging MVDream [67] as a diffusion prior that has been pre-trained on large-scale image and 3D datasets.

3 Preliminaries

We introduce some of the preliminary concepts that are needed for the later discussion. In Sec. 3.1, we provide an overview of 3D meshes and their relevant attributes. Then, we provide a tutorial about diffusion models in Sec. 3.2. We present the concept of SDS in Sec. 3.3 and provide an overview of the criteria needed for a mesh to be operational in Sec. 3.4.

3.1 3D Meshes

In this section, we provide a mathematical definition of the *components* that compose three-dimensional (3D) meshes. A mesh is a polyhedral representation of the surface of an object. As such, the mesh offers a high degree of flexibility and can be extended arbitrarily with additional components that can describe physical or even semantic properties of the modeled objects. Here, we focus on the parts of a mesh that are widely adopted and used in the entertainment sector to produce Computer-Generated Imagery (CGI). We present a general description of the 3D mesh that is compatible with standard mesh data structures, file formats, mesh processing algorithms, and rendering and shader pipelines. In Section 3.4, we discuss the *properties* that a mesh needs to satisfy to be operational for downstream applications in the entertainment industry.

3.1.1 Shape and Structure

Geometry. The geometry of the mesh is the cornerstone of the 3D mesh architecture, dictating the spatial manifestation of the modeled objects. Let $\boldsymbol{V}$ be the vertex matrix such that $\boldsymbol{V} \in \mathbb{R}^{N\times 3}$, where each row $\boldsymbol{v}_i = (x_i, y_i, z_i)$ represents the coordinates of the i-th vertex in three-dimensional space, for $i = 1, \ldots, N$. Practically, $\boldsymbol{V}$ represents a point cloud.

Topology. The topology describes how these vertices interconnect, effectively controlling the deformation characteristics and structural integrity of the mesh. Let $\boldsymbol{E}$ be the edge matrix such that $\boldsymbol{E} \in \{1, \ldots, N\}^{M\times 2}$, where each row $\boldsymbol{e}_k = (v_{1_k}, v_{2_k})$ contains indices v_{1_k} and v_{2_k} of $\boldsymbol{V}$, denoting the two vertices that form the k-th edge, for $k = 1, \ldots, M$. Equivalently, the connectivity of the mesh is also represented by the face matrix $\boldsymbol{F}$ such that $\boldsymbol{F} \in \{1, \ldots, N\}^{P\times 3}$, where each row $\boldsymbol{f}_l = (v_{1_l}, v_{2_l}, v_{3_l})$

consists of vertex indices $v_{1_l}, v_{2_l}, v_{3_l}$ of $\boldsymbol{V}$, delineating the vertices that define the l-th triangular face[1], for $l = 1, \ldots, P$.

3.1.2 Surface Properties

Materials. Every face in $\boldsymbol{F}$ has one material assigned to it. A material in 3D graphics is a collection of properties that determine how a set of faces on a mesh will appear when rendered. These properties define the color and how the surface interacts with light. Common material properties can include:

- Albedo (Diffuse): The base color of the material.
- Specularity: How shiny the surface is and the color of its shine.
- Roughness: How rough or smooth the surface is, affecting how sharp or diffuse the reflections are.
- Metalness: Whether the surface behaves more like a metal or a non-metal.
- Transparency: How transparent the surface is.
- Emission: Whether the surface emits light.

A material is tied to a shader, which is a program that runs on the graphics hardware to actually render the surface with all its defined properties.

Texture maps. Each of the properties in the material can be controlled by a texture map, which allows for more complex and varied surface appearances than could be achieved with uniform values alone. Texture maps are 2D images that provide spatially varying information to the shaders to control specific properties of the material across the surface of a 3D object. Depending on the material property, the texture map might be stored either as a grayscale or as an RGB image.

UV coordinates. Texture maps are linked to the surface through UV coordinates, which tell the shader how to wrap the 2D texture data onto the 3D surface in a process known as *UV mapping*. Let $\boldsymbol{W}$ be the UV coordinate matrix such that $\boldsymbol{W} \in [0, 1]^{N \times 2}$, where each row $\boldsymbol{w}_p = (u_p, v_p)$ represents the 2D texture coordinates of the p-th point in UV space, for $p = 1, \ldots, Q$, where the number of UV coordinates Q is greater or equal to the number of vertices N (i.e., $Q \geq N$). In the simplest case, where $Q = N$, there is one UV coordinate per vertex.

[1] Note that faces in a mesh can be modeled as any planar polygon of degree greater than 2, such as quadrangles, which are prevalent in applications requiring mesh deformations. However, we assume triangular faces in this context, since any mesh can be reversibly triangulated. Triangular faces offer the advantage of guaranteed planarity, simplifying the mathematical modeling and rendering processes while ensuring stability and compatibility across different 3D graphics systems.

Vertex texture coordinate indices. However, it is often the case that one vertex is mapped to different UV coordinates if that vertex is indexed by multiple faces. In this case, there are more UV coordinates than there are vertices, i.e., $Q > N$, which is especially useful to avoid visible seams when UV mapping a texture that contains texture islands. We denote the texture coordinate index matrix as $\boldsymbol{T}$ such that $\boldsymbol{T} \in \{1, \ldots, Q\}^{P \times 3}$, where each row $\boldsymbol{t}_l = (w_{1_l}, w_{2_l}, w_{3_l})$ contains indices of $\boldsymbol{W}$, corresponding to the texture UV coordinates associated with the vertices of the l-th face in the mesh, for $l = 1, \ldots, P$.

UDIM tiles. A UDIM system is a collection of several 2D texture spaces (tiles), each identified by a unique index. The UDIMs extend the traditional single UV tile system by mapping UV coordinates across multiple tiles, increasing texture resolution and detail. In a UDIM setup, the UV coordinates are redefined to a larger range $\boldsymbol{W}^{\text{UDIM}} \in [0, \infty)^{N \times 2}$ that spans multiple texture tiles. The tile number for each UV coordinate is given by:

$$\text{Tile Index} = 1001 + \lfloor u_p \rfloor + 10 \lfloor v_p \rfloor$$

where $\lfloor \cdot \rfloor$ denotes the floor function, which maps the decimal UV coordinates into integer indices corresponding to specific texture tiles. Here, 1001 is the starting index for the first tile.

3.1.3 Deformation Primitives

Rigging. A mesh can can be animated via rigging, which is the process of assigning a hierarchical skeletal structure of bones to the mesh. Then, a set of skinning weights is defined on to control how much each bone influences the position of each vertex in the mesh. The deformation is commonly computed via linear blend skinning or dual quaternion skinning [31].

Blend shapes. Alternatively (or in combination with the rig), a mesh can also be animated via blend shapes or morphs, which are meshes that share the same topology. Blend shapes are often useful to control detail in animations, for example around facial animation, where a skeletal rig would not be suited for practical considerations.

Physical properties. Other physical surface properties can be defined to deform the mesh via physics simulation. This can include vertex attributes such as the mass or restitution coefficients, as well as information that is used to compute collisions.

3.2 Diffusion Models

Denoising diffusion probabilistic models [22, 53] are a class of generative models that have become the state-of-the-art in several downstream applications, such as image generation [58]. The general idea is to iteratively remove noise from a multivariate data point. An interpretation of diffusion modeling is that starting with isotropic noise, we use Langevin dynamics via the gradients of the data distribution approximated with score matching to sample new points that lie in the lower dimensional data manifold [70]. The main difficulty lies in the complexity of the data distribution, where even high-density areas are extremely scarce, especially in the case of high dimensionality.
Consider the D-dimensional data point $\mathbf{x}_0 \in \mathbb{R}^D$, which is sampled from the unknown data distribution $q(\mathbf{x})$ as $\mathbf{x}_0 \sim q(\mathbf{x})$. Diffusion models aim to approximate the real data distribution $q(\mathbf{x})$ via the probability distribution $p_{\boldsymbol{\theta}}(\mathbf{x})$, parameterized by $\boldsymbol{\theta}$ as a neural network, such that $\mathbf{x}_0$ can be sampled along with new data points. Thus, the model is trained to minimize the negative log-likelihood

$$\mathcal{L} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x})} \left[-\log \left(p_{\boldsymbol{\theta}} (\mathbf{x}_0) \right) \right]. \tag{3.1}$$

However, the loss function (3.1) is intractable, because we define the marginal distribution $p_{\boldsymbol{\theta}}(\mathbf{x}_0)$ over the data point $\mathbf{x}_0$ as

$$p_{\boldsymbol{\theta}} (\mathbf{x}_0) := \int p_{\boldsymbol{\theta}} (\mathbf{x}_{0:K}) \, \mathrm{d}\mathbf{x}_{1:K} \tag{3.2}$$

by integrating the joint probability distribution $p_{\boldsymbol{\theta}}(\mathbf{x}_{0:K})$ over the latent random variables $\mathbf{x}_1, \dots \mathbf{x}_K$, where $\mathbf{x}_k \in \mathbb{R}^D \; \forall k \in \{1, \dots, K\}$ and $K \in \mathbb{N}$. In contrast to other generative deep latent-variable models, such as variational autoencoders (VAEs, [35]), the posterior probability distribution

$$q (\mathbf{x}_{1:K} \mid \mathbf{x}_0) := \prod_{k=1}^{K} q (\mathbf{x}_k \mid \mathbf{x}_{k-1})$$

over the sequence of latents has no learnable parameters. Instead, we obtain the latent variables by sequentially applying a degradation operation to the original signal $\mathbf{x}_0$. The so-called *forward diffusion process* consists of a Markov chain that gradually adds Gaussian noise to the input over K steps with the transition probability distribution

$$q (\mathbf{x}_k \mid \mathbf{x}_{k-1}) = \mathcal{N} \left(\mathbf{x}_k; \sqrt{1 - \beta_k} \mathbf{x}_{k-1}, \beta_k \mathbf{I} \right). \tag{3.3}$$

The monotonically increasing forward noising schedule $\beta_1, \dots \beta_K$, where $\beta_k \in (0, 1) \; \forall k \in \{1, \dots, K\}$, controls the variance of the noise contamination. Using

the reparametrization trick, we find a closed-form representation to efficiently sample the latent

$$\mathbf{x}_k \sim q\left(\mathbf{x}_k \mid \mathbf{x}_0\right) = \mathcal{N}\left(\mathbf{x}_k; \sqrt{\bar{\alpha}_k}\mathbf{x}_0, \left(1-\bar{\alpha}_k\right)\mathbf{I}\right) \tag{3.4}$$

at any step $k \in \{0, \ldots, K\}$ as an interpolation $\mathbf{x}_k = \sqrt{\bar{\alpha}_k}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_k}\boldsymbol{\varepsilon}$ between the clean signal $\mathbf{x}_0$ and some fixed Gaussian noise $\boldsymbol{\varepsilon} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{I}\right)$ with $\alpha_k := 1 - \beta_k$ and $\bar{\alpha}_k := \prod_{i=1}^{t} \alpha_i$. Typically, we design the noising schedule so that $\bar{\alpha}_K \approx 0$, by which

$$q\left(\mathbf{x}_K\right) := \int q\left(\mathbf{x}_K \mid \mathbf{x}_0\right) q\left(\mathbf{x}_0\right) \mathrm{d}\mathbf{x}_0 \approx \mathcal{N}\left(\mathbf{x}_K; \mathbf{0}, \mathbf{I}\right)$$

can be assumed to be an isotropic Gaussian for $K \to \infty$. Nichol et al. [53] propose using a fixed cosine schedule[2] with $K = 1000$ for the forward process. Then, the *reverse diffusion process*, described by the joint probability distribution $p_{\boldsymbol{\theta}}\left(\mathbf{x}_{0:K}\right)$ from (3.2), can also be modeled as a Markov chain

$$p_{\boldsymbol{\theta}}\left(\mathbf{x}_{0:K}\right) := p\left(\mathbf{x}_K\right) \prod_{k=1}^{K} p_{\boldsymbol{\theta}}\left(\mathbf{x}_{k-1} \mid \mathbf{x}_k\right) \tag{3.5}$$

that starts with a sample $\mathbf{x}_K$ of pure noise drawn from the prior distribution $p\left(\mathbf{x}_K\right) = \mathcal{N}\left(\mathbf{x}_K; \mathbf{0}, \mathbf{I}\right)$ as $\mathbf{x}_K \sim p\left(\mathbf{x}_K\right)$. Therefore, the network must learn Gaussian transition kernels that approximate the inverse $q\left(\mathbf{x}_{k-1} \mid \mathbf{x}_k\right)$ of the forward transitions in (3.3) as

$$p_{\boldsymbol{\theta}}\left(\mathbf{x}_{k-1} \mid \mathbf{x}_k\right) := \mathcal{N}\left(\mathbf{x}_{k-1}; \boldsymbol{\mu}_{\boldsymbol{\theta}}\left(\mathbf{x}_k, k\right), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}\left(\mathbf{x}_k, k\right)\right). \tag{3.6}$$

The choice of Gaussian transitions in (3.6) is valid as long as β_k is kept small [68]. This way, the model needs to learn the parameters $\boldsymbol{\mu}_{\boldsymbol{\theta}}\left(\mathbf{x}_k, k\right)$ and $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}\left(\mathbf{x}_k, k\right)$ of the reverse transition. In this sense, we can approximate the log-likelihood objective $\mathcal{L}$ in 3.1 via the evidence lower bound (ELBO)

$$\mathcal{L} \leq \mathcal{L}_K + \sum_{k=2}^{K} \mathcal{L}_{k-1} + \mathcal{L}_0 \tag{3.7}$$

$$\mathcal{L}_K = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x})}\left[D_{\mathsf{KL}}\left(q\left(\mathbf{x}_K \mid \mathbf{x}_0\right) \parallel p\left(\mathbf{x}_K\right)\right)\right] \tag{3.7a}$$

$$\mathcal{L}_{k-1} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}), \mathbf{x}_k \sim q(\mathbf{x}_k \mid \mathbf{x}_0)}\left[D_{\mathsf{KL}}\left(q\left(\mathbf{x}_{k-1} \mid \mathbf{x}_k, \mathbf{x}_0\right) \parallel p_{\boldsymbol{\theta}}\left(\mathbf{x}_{k-1} \mid \mathbf{x}_k\right)\right)\right] \quad \forall k \in \{2, \ldots, K\} \tag{3.7b}$$

$$\mathcal{L}_0 = -\mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}), \mathbf{x}_1 \sim q(\mathbf{x}_1 \mid \mathbf{x}_0)}\left[\log\left(p_{\boldsymbol{\theta}}\left(\mathbf{x}_0 \mid \mathbf{x}_1\right)\right)\right] \tag{3.7c}$$

The term (3.7a) has no learnable parameters $\boldsymbol{\theta}$ and is thus ignored during training, although it is close to zero if the requirement $q\left(\mathbf{x}_K \mid \mathbf{x}_0\right) \approx \mathcal{N}\left(\mathbf{x}_K; \mathbf{0}, \mathbf{I}\right)$ is fulfilled.

[2] Recent work [19, 42] shows that commonly used variance schedules fail at completely removing lower-frequency information from the clean data point. Unless zero signal-to-noise ratio is enforced at the end of the forward diffusion process, the reverse process suffers from a distribution shift between the aggregate posterior $q\left(\mathbf{x}_K \mid \mathbf{x}_0\right)$ seen during training and the prior $p\left(\mathbf{x}_K\right)$ used for generation.

Also, (3.7c) represents a variational reconstruction term. Notably, the forward process posteriors $q\left(\mathbf{x}_{k-1} \mid \mathbf{x}_k\right)$ become tractable when conditioned on $\mathbf{x}_0$ via Bayes theorem, such that we can use the KL divergence in (3.7b) to compute the variational gap between $p_{\boldsymbol{\theta}}\left(\mathbf{x}_{k-1} \mid \mathbf{x}_k\right)$ and

$$q\left(\mathbf{x}_{k-1} \mid \mathbf{x}_k, \mathbf{x}_0\right) = \mathcal{N}\left(\mathbf{x}_{k-1}; \tilde{\boldsymbol{\mu}}_k\left(\mathbf{x}_k, \mathbf{x}_0\right), \tilde{\beta}_k \mathbf{I}\right),$$

where the mean $\tilde{\boldsymbol{\mu}}_k\left(\mathbf{x}_k, \mathbf{x}_0\right)$ and the variance schedule $\tilde{\beta}_k$ are given by

$$\begin{aligned} \tilde{\boldsymbol{\mu}}_k\left(\mathbf{x}_k, \mathbf{x}_0\right) &:= \frac{\sqrt{\bar{\alpha}_{k-1}}\beta_k}{1-\bar{\alpha}_k}\mathbf{x}_0 + \frac{\sqrt{\alpha_k}\left(1-\bar{\alpha}_{k-1}\right)}{1-\bar{\alpha}_k}\mathbf{x}_k, \\ \tilde{\beta}_k &:= \frac{1-\bar{\alpha}_{k-1}}{1-\bar{\alpha}_k}\beta_k. \end{aligned} \tag{3.8}$$

The variance $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}\left(\mathbf{x}_k, k\right)$ in the reverse transition kernel (3.6) can be either fixed [22] to $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}\left(\mathbf{x}_k, k\right) := \sigma_t^2 \mathbf{I}$ or made learnable [53]. The mean $\boldsymbol{\mu}_{\boldsymbol{\theta}}\left(\mathbf{x}_k, k\right)$ could be directly predicted, but we obtain better results by substituting the explicit form of $\mathbf{x}_k$ from (3.4) into the formulation of the mean in (3.8) to obtain

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}\left(\mathbf{x}_k, k\right) = \frac{1}{\sqrt{\alpha_k}}\left(\mathbf{x}_k - \frac{1-\alpha_k}{\sqrt{1-\bar{\alpha}_k}}\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}\left(\mathbf{x}_k, k\right)\right)$$

with the neural network $\boldsymbol{\varepsilon}_{\boldsymbol{\theta}}\left(\mathbf{x}_k, k\right)$ that predicts the noise $\boldsymbol{\varepsilon}$ present in $\mathbf{x}_k$. The ELBO in (3.7) can then be reweighted by sampling k from the uniform distribution $k \sim \mathcal{U}\left(1, K\right)$ and computing the denoising term $\mathcal{L}_{k-1}$ from (3.7b) for $k \in \{1, \ldots, K\}$, which can be simplified to

$$\mathcal{L}_{\text{simple}} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}), k \sim \mathcal{U}(1,K), \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})}\left[\left\|\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}_{\boldsymbol{\theta}}\left(\mathbf{x}_k, k\right)\right\|^2\right] \tag{3.9}$$

in the case of fixed variance $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}\left(\mathbf{x}_k, k\right) := \sigma_t^2 \mathbf{I}$.
The neural network $\boldsymbol{\theta}$ is commonly parameterized as a U-Net [59] with self-attention [3, 49]. Rombach et al. [58] propose to denoise the data points in the latent space of a VAE to alleviate the high computational and time cost of diffusion models both during training and at sampling time. Moreover, more efficient sampling techniques, modified formulations of the simplified objective, and distillation approaches have been proposed in [56, 62, 69] and widely adopted by the research community. Classifier-free guidance (CFG) [23] has become a popular method to condition diffusion models on feature vectors, such as CLIP [57] embeddings of text descriptions. The guidance parameter can be tuned to achieve a balance between perceptual quality of the generations (often measured in terms of the FID score [21]) and variation between samples. We explore CFG in more depth in Sec. 3.2.1. Furthermore, the ControlNet [78] architecture allows to spatially condition the diffusion model and fine-tune large pre-trained models. There have been

claims made that Gaussian noise is not necessary for denoising diffusion models to work and that the model can work even with deterministic perturbations [6], showing the robustness of the diffusion framework. However, non-Gaussian perturbations have been found to hurt the generative performance both in terms of quality and variability.

3.2.1 Classifier-free Guidance

Classifier-free guidance, introduced by [23], addresses limitations associated with classifier guidance in diffusion models. Traditional methods often rely on a separate classifier to guide the generation process, which can introduce biases and constraints due to the classifier's training.
In contrast, classifier-free guidance eliminates the need for a separate classifier by integrating guidance directly into the diffusion model. This is achieved by modifying the training objective to include a conditioning variable $\boldsymbol{y}$, allowing the model to learn both conditional and unconditional distributions. During inference, the generation process can be guided by interpolating between these distributions, effectively controlling the trade-off between sample quality and adherence to the conditioning variable $\boldsymbol{y}$. The equation for CFG is given by:

$$\hat{\boldsymbol{\varepsilon}}_{\boldsymbol{\phi}}(\boldsymbol{x}_t; \boldsymbol{y}, t) = (1 + w_{\text{CFG}})\boldsymbol{\varepsilon}_{\boldsymbol{\phi}}(\boldsymbol{x}_t; \boldsymbol{y}, t) - w_{\text{CFG}}\boldsymbol{\varepsilon}_{\boldsymbol{\phi}}(\boldsymbol{x}_t; t), \tag{3.10}$$

where $\hat{\boldsymbol{\varepsilon}}_{\boldsymbol{\phi}}(\boldsymbol{x}_t; \boldsymbol{y}, t)$ is the predicted noise, $\boldsymbol{\varepsilon}_{\boldsymbol{\phi}}(\boldsymbol{x}_t; \boldsymbol{y}, t)$ denotes the conditional noise, $\boldsymbol{\varepsilon}_{\boldsymbol{\phi}}(\boldsymbol{x}_t; t)$ represents the unconditional noise, and w_{CFG} is the CFG weight. The unconditional model is conditioned on the null token $\emptyset$ as the unconditional class identifier, meaning that $\boldsymbol{\varepsilon}_{\boldsymbol{\phi}}(\boldsymbol{x}_t; t) := \boldsymbol{\varepsilon}_{\boldsymbol{\phi}}(\boldsymbol{x}_t; \boldsymbol{y} = \emptyset, t)$. At training, we jointly learn a conditional and an unconditional model by randomly dropping the conditioning signal $\boldsymbol{y}$ according to some probability p_{uncond}, which is a hyperparameter. We consider CFG to be an elegant method due to the simplicity of the approach, where the same U-Net [59] can be trained to perform both tasks (conditional and unconditional prediction) and there is no need for an additional classifier, avoiding classifier-based adversarial behavior.

3.3 Score Distillation Sampling

Score Distillation Sampling (SDS) [56] is a novel technique developed to improve the generation quality of 3D generative models and the tractability in the context of diffusion models. It leverages the foundational principles of diffusion models and introduces a unique optimization-based approach to sampling that operates in the parameter space rather than the pixel space, enabling effective text-to-3D synthesis.
In standard diffusion models, ancestral sampling is typically used, where the model

generates data samples directly in the pixel space. However, SDS diverges from this approach by optimizing over the parameters of a differentiable image parameterization, such as a NeRF [50], which maps these parameters to the generated image. SDS begins with a randomly initialized parameter vector z. These parameters could define a 3D volume, which when rendered, generates the desired image. Instead of generating samples in the pixel space, SDS optimizes the parameters z such that the resulting image $\boldsymbol{x} = \mathbf{r}(z)$, where $\mathbf{r}(z)$ is a forward model, appears as if it were a sample from the diffusion model. This is done by diffusing the image and using the diffusion model's score function to guide the optimization process. The key to SDS is its loss function, which is inspired by probability density distillation. The score distillation loss $\mathcal{L}_{\text{SDS}}$ is designed to minimize the Kullback-Leibler (KL) divergence between the Gaussian distribution family and the score functions learned by the pre-trained diffusion model. The gradient of the SDS objective is given by:

$$\nabla_\theta \mathcal{L}_{\text{SDS}}(\boldsymbol{\phi}, \boldsymbol{x} = \mathbf{r}(z)) = \mathbb{E}_{t,\boldsymbol{\varepsilon}} \left[w(t) \left(\hat{\boldsymbol{\varepsilon}}_{\boldsymbol{\phi}}(\boldsymbol{x}_t; y, t) - \boldsymbol{\varepsilon} \right) \frac{\partial \boldsymbol{x}_t}{\partial z} \right] \tag{3.11}$$

This loss function effectively guides the optimization towards regions of the parameter space that correspond to high-density regions of the data distribution.
The gradient of the score distillation loss is computed without backpropagating through the diffusion model's U-Net, simplifying the optimization process. This is achieved by treating the diffusion model as a frozen critic that provides a direction for image-space edits without the need for backpropagation through the entire model.
SDS has been applied successfully in generating high-fidelity 3D models from text descriptions. By combining SDS with a NeRF-like model tailored for 3D generation, methods like DreamFusion [56] have been able to produce coherent 3D objects and scenes from natural language prompts. This approach does not require 3D or multi-view training data, making it highly versatile and effective for various generative tasks.
Score Distillation Sampling represents a significant advancement in the field of generative modeling, particularly for applications requiring the generation of complex, high-dimensional data such as 3D models. By leveraging the strengths of diffusion models and introducing an innovative parameter space optimization approach, SDS enables the creation of high-quality generative outputs with improved fidelity and coherence.

3.4 Defining Qualitative Operability Criteria for 3D Meshes

In this section, we define the minimum requirements *sine qua non* for a 3D mesh to be **operational** or production-ready for downstream industry applications. Meshes are commonly employed in the design, manufacturing, simulation, architecture, film, game, and media industries.

Geometry and topology. Geometry, encompassing vertices, edges, and faces, forms the cornerstone of 3D mesh architecture, dictating the spatial manifestation of objects. Topology, the study of how these elements interconnect, further refines this by informing on the mesh's deformation characteristics and structural integrity. For a mesh to be deemed production-ready, its topology should predominantly consist of quadrilaterals, facilitating smoother deformations and subdivisions, crucial for high-fidelity animations. Efficiency and consistency in mesh density, tailored to the object's visual prominence and interaction within digital environments, are paramount. It is also beneficial to have anisotropy in the topology to enable regions of high detail, and edge loops should align with semantic features.

Surface definition. The surface definition, characterized by normals, materials, and UV maps, serves as the nexus between the mesh's geometric form and its visual perception. Normals influence light reflection, essential for realistic rendering. Correct orientation and calculation of normals are vital for mimicking natural light interactions. Materials, through the specification of physical properties (e.g., albedo, roughness), and UV maps, by dictating texture placement, collectively enrich the mesh with life-like detail. Production-readiness demands materials compatible with physically based rendering (PBR) standards and UV maps optimized for minimal distortion, ensuring the verisimilitude of textures on complex surfaces. Consistency is also a key feature of UV maps, as a shared UV space can be used to interchange textures between different assets. Moreover, UVs should follow the structures defined by the topology to hide visible seams along texture islands.

Rigging. It is critical that meshes share a common rig, such that shared animation libraries can be accessed. The skeleton must have accurately placed bones and cover all joints present in the mesh. Skinning weights must be optimized to work with a specific skinning algorithm and provide control over all bones in the skeleton with a high degree of accuracy. The design of the skinning weights must take into account the underlying topology, which defines the deformation behavior of the mesh surface.

3.4.1 Coherence and consistency.

Lastly, the mesh's integration within its intended setting mandates attention to stylistic coherence, and environmental consistency. For instance, in the context of meshes that represent humanoid figures, anatomical realism becomes an important factor.
By establishing these criteria and components as the bedrock for 3D mesh development, this discourse aims to set a clear, comprehensive stage for subsequent examinations into our method.

4 Method

In this chapter, we present our generative model of humanoid meshes. We provide an overview of our approach in Sec. 4.1 and discuss each of the components of our algorithm in Secs. 4.2 - 4.4.

4.1 Overview

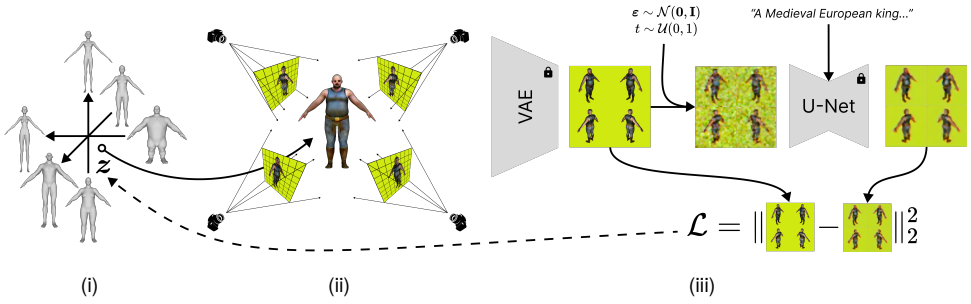


Figure 4.1: Overview of our method, which consists of three main components. The geometric prior (i) defines the shape space of a 3DMM with known topology, skinning weights, and UVs. We sample a mesh from the latent code z. Next, we render the mesh from different views with shaded textures (ii). At last, the mutliview diffusion prior (iii) provides the objective $\mathcal{L}$ that is backpropagated all the way back to z, as indicated by the dashed arrow. We encode the renderings with a pre-trained, frozen VAE and perturb the latents with isotropic Gaussian noise $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and noise scale $t \sim \mathcal{U}(0, 1)$. We denoise the noised latents with a multi-view diffusion U-Net conditioned on a text prompt. The loss is equivalent to the squared Euclidean norm of the reconstruction error. We use the icon of a lock to point out networks with non-trainable weights.

SDS methods consist of a forward model that samples images by projecting some underlying 3D neural representation (usually some variant of NeRF [50] or Gaussian Splatting [32]) onto the image plane, and a diffusion prior that acts as a critic by denoising the renderings after being perturbed with Gaussian noise. The error gradient is then backpropagated into the 3D representation and the process is repeated iteratively until convergence. As a last step, a mesh is extracted from the 3D representation, usually via Marching Cubes [46].
Our motivation is to introduce a geometric prior that operates in mesh space into the forward model. The reasons are that (i) a geometric prior in mesh space allows

to better control the structural properties of the output mesh, which is not possible with Marching Cubes or post-processing algorithms. We can directly sample the output mesh from the geometric prior. Also, (ii) the geometric prior is a strong prior over the distribution of shapes and appearance, regularizing the diffusion and helping in turn the mode seeking nature of SDS. And finally, (iii) sampled shapes correspond to an identity code that is interpretable, controllable, and editable
We construct the geometric prior as a 3D morphable model that defines a shape space as an orthogonal and uncorrelated subspace of our own dataset of blend shapes of humanoids. The basis of the shape space is spanned by a set of PCs, each corresponding to an eigenvector of the covariance matrix derived from the data. In the context of a mesh-based representation, each PC defines a specific mode of variation around the template mesh. These modes can be interpreted as displacements from the average mesh. The sign and magnitude of the coefficient for each PC determine the direction and extent of this deformation from the average shape. Thus, the geometric prior serves both as a prior of the geometric properties of the output mesh (topology, skinning weights, texture coordinates, vertex attributes, etc.), and as a shape prior, where each point in the shape space represents a unique shape.
In essence, our generative framework performs an iterative search over the shape space of the 3DMM and the appearance space defined by a neural field to find the 3D object that best matches an input textual description. In every iteration, we use a differentiable renderer to render the mesh into a set of images from four orthogonal perspectives. Then, a multi-view diffusion prior acts as a visual critic that provides guidance via gradient descent. We provide an overview of the proposed approach in Fig. 4.1.
The remainder of this chapter is structured as follows: we introduce the geometric prior in Sec. 4.2. Next, we discuss the differentiable rendering mechanism that completes the forward model in Sec. 4.3. We close by presenting the diffusion prior in Sec. 4.4.

4.2 Geometric Prior

In this section, we focus on the development of a shape generative model capable of capturing geometric variations across different samples while maintaining a fixed topology. By *shape*, we specifically refer to the vertex positions of a triangular mesh, characterized by some given mesh connectivity. We define the generator as a point distribution model, which is a function

$$\mathbf{g}_{\boldsymbol{\phi}} : \mathcal{Z} \subseteq \mathbb{R}^K \rightarrow \mathbb{R}^{3N}, \tag{4.1}$$

parameterized by $\boldsymbol{\phi}$, which maps a latent vector $\boldsymbol{z} \in \mathcal{Z}$ to a shape $\boldsymbol{s} \in \mathbb{R}^{3N}$ as $\boldsymbol{s} = \mathbf{g}_{\boldsymbol{\phi}}(z)$. Hence, we aim to approximate with $\mathbf{g}_{\boldsymbol{\phi}}(z)$ the prior probability den-

sity $\mathbf{q}(z)$ over the shape space $\mathcal{Z}$. This prior probability distribution serves as a measure of the likelihood that a given vector z within the shape space represents a realistic and plausible shape s. Several methodologies exist for constructing such shape generators, including non-linear models like variational autoencoders (VAEs) [33, 35] or autodecoders (ADs) [50]. In our own experience, VAEs are more flexible than the data that they are trained on and learn behavior that is more complex, learning to autoencode and generate data patterns that they have not been trained on. While this inherent versatility of AEs can be beneficial for applications such as image compression, it is hard to constrain an AE to accurately model the shape space of a dataset of meshes without undesired hallucinations of out-of-distribution shapes.
To enhance the interpretability of our generative framework, we opt for 3DMMs as our geometric prior, which leverages linear statistical shape analysis as the foundation of its generative process. 3DMMs are a type of statistical model used to represent and manipulate the 3D shapes and appearances of objects, most commonly human faces. The core idea behind the seminal work of 3DMMs [8] is that any face can be represented as a linear combination of the mean face and a set of principal components derived from the training data. These components capture key variations, such as differences in facial structure, expressions, and skin textures. Practically, 3DMMs are the extension of the concept of eigenfaces [74] from 2D images to 3D meshes. The 3DMM can synthesize realistic and unseen faces by linearly interpolating or extrapolating samples of the training set. The same idea can be extended beyond faces to more general shapes, like the full human body [14, 45] or even quadruped animals [79].

4.2.1 3D Morphable Model

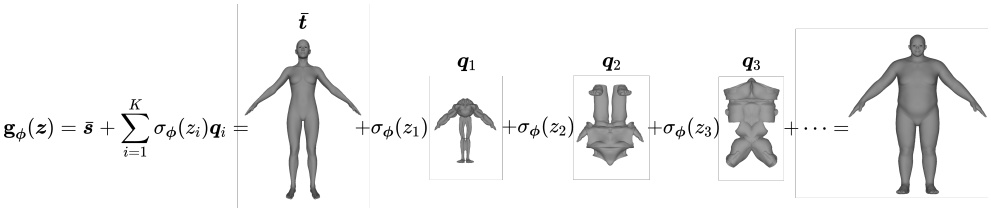


Figure 4.2: Illustration of the differentiable generator $\mathbf{g}_{\boldsymbol{\phi}}(z)$, which maps a parameter vector z in a K-dimensional shape space to an output mesh as a linear combination of PC blend shapes. Each PC $\boldsymbol{V}\mathbf{e}_i$ defines a specific mode of variation around the template shape $\bar{s}$. These modes can be interpreted as displacements from the average mesh (for visibility, we scaled up the PCs in the depiction, as their magnitude is very close to the origin). The linear coefficients $\sigma_{\boldsymbol{\phi}}(z_i)$, which constrain the shape space to the AABB that is defined by the dataset projected to the PCA space, modulate the contribution of each PC.

Given a dataset of 3D blend shapes with fixed topology, our goal is to construct

a 3DMM by performing Principal Component Analysis (PCA) over the differences of the vertices with respect to a template shape. Let the dataset

$$\mathbf{D} = [\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_m] \tag{4.2}$$

consist of $|\mathbf{D}|$ blend shapes, each containing n vertices. Let each blend shape be represented as a vector $\boldsymbol{s}_i \in \mathbb{R}^{3n}$ for $i = 1, 2, \ldots, |\mathbf{D}|$. We begin by defining the template shape $\bar{\boldsymbol{s}}$ as the average of the dataset. The template shape is computed as:

$$\bar{\boldsymbol{s}} = \frac{1}{|\mathbf{D}|} \sum_{i=1}^{|\mathbf{D}|} \boldsymbol{s}_i. \tag{4.3}$$

Next, we center the data by calculating the difference vector between each blend shape $\boldsymbol{s}_i$ and the average shape $\bar{\boldsymbol{s}}$:

$$\boldsymbol{\Delta s}_i = \boldsymbol{s}_i - \bar{\boldsymbol{s}}, \quad \text{for } i = 1, 2, \ldots, |\mathbf{D}|. \tag{4.4}$$

We organize the difference vectors into a data matrix $\boldsymbol{\Delta}\mathbf{S} \in \mathbb{R}^{3n \times |\mathbf{D}|}$ as follows:

$$\boldsymbol{\Delta}\mathbf{S} = \left[\boldsymbol{\Delta s}_1, \boldsymbol{\Delta s}_2, \ldots, \boldsymbol{\Delta s}_{|\mathbf{D}|}\right]. \tag{4.5}$$

Because of the centering in Eq. (4.4), the data matrix $\boldsymbol{\Delta}\mathbf{S}$ has column-wise zero empirical mean. Each column represents the deviation of a blend shape relative to the template[1]. We compute the covariance matrix $\mathbf{C}$ of the centered data:

$$\mathbf{C} = \frac{1}{|\mathbf{D}| - 1} \boldsymbol{\Delta}\mathbf{S}\boldsymbol{\Delta}\mathbf{S}^\top, \tag{4.6}$$

where $\mathbf{C} \in \mathbb{R}^{3n \times 3n}$. The covariance matrix captures the relationships between the deviations of the vertices in the blend shapes. We perform an eigen decomposition of the covariance matrix to obtain the eigenvalues λ_i and eigenvectors $\boldsymbol{q}_i$:

$$\mathbf{C}\boldsymbol{q}_i = \lambda_i \boldsymbol{q}_i, \quad \text{for } i = 1, 2, \ldots, \mathbf{min}(3n, |\mathbf{D}|). \tag{4.7}$$

The eigenvectors $\boldsymbol{q}_i$ represent the principal components (directions) of the variation in the dataset, and the corresponding eigenvalues λ_i indicate the amount of variance explained by each component. The number of non-zero eigenvalues is limited by the smaller dimension of the data matrix, which is the minimum between the number of vertices $(3N)$ or the number of blend shapes $(|D|)$, indicated by $\mathbf{min}(3N, |\mathbf{D}|)$. The 3D morphable model is then constructed using the template

[1] Note that common PCA applications, such as feature or dimensionality reduction, stack different data samples along the row axis. Instead, we aim to extract orthonormal principal components of the data as a way of "data reduction". Therefore, we stack the data samples along the column dimension.

shape $\bar{\boldsymbol{s}}$, the PCs $\boldsymbol{q}_i$, and the eigenvalues λ_i. A new shape $\boldsymbol{s}_{\text{new}}$ can be generated as:

$$\boldsymbol{s}_{\text{new}} = \bar{\boldsymbol{s}} + \sum_{i=1}^{K} \omega_i \boldsymbol{q}_i, \tag{4.8}$$

where ω_i are the coefficients associated with the orthonormal PCs, and K is the number of PCs retained to approximate the shape. We substitute ω_i with another variable in Sec. 4.2.2. The choice of K depends on the desired balance between model complexity and fidelity to the original data. By applying PCA to the deviations of blend shapes from a template, we have derived a compact representation of the variability inherent in the dataset. This model allows for the generation of new, realistic 3D shapes that adhere to the patterns of variation observed in the original blend shapes. The PCs provide a reduced-dimensionality basis for representing shape variation, facilitating efficient manipulation and synthesis of new shapes in 3D space.

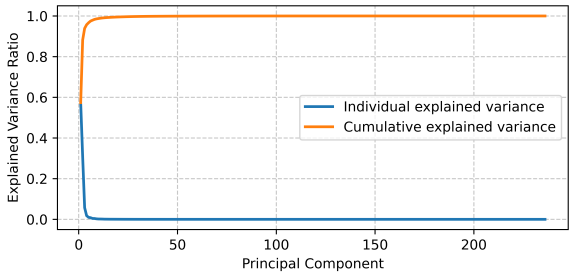


Figure 4.3: We perform PCA over the dataset and plot the individual and cumulative explained variance by all PCs. We observe a steep increase of the cumulative explained variance: 99% of the variance is contained in the first 12 PCs.

Despite the fact that only the first 12 PCs are needed to explain 99% of the variance in the data (see Fig. 4.3), we still choose the first $K = 100$ PCs to construct the shape space of our geometric prior. We choose such a high cutoff mainly for two reasons:

1. In principle, the cardinality of our dataset ($|\mathbf{D}| = 236$, see Sec. 4.2.3) is small enough that the original samples could be used for interpolation without performing dimensionality reduction via PCA. However, performing PCA improves the optimization landscape of the shape space for the subsequent gradient descent optimization, because the PCs form an orthogonal and uncorrelated basis of the dataset with diagonalized covariance matrix. Thus, each parameter (corresponding to a PC) can be optimized independently without affecting the others, i.e., the optimization process does not have to

deal with cross-coupled parameters. The simplified landscape is smoother and less prone to local minima, as every point in the space is guaranteed to be unique. We ablate the choice of the PCA shape space in Sec. 5.7.

2. Additionally, high-frequency, small-scale facial details, such as wrinkles or subtle curvature of the nose, are critical for human perception but may account for a minor portion of the overall variance in the data, as they are dominated by low-frequency, large-scale variations like character height or width. Therefore, we include a larger number of PCs to ensure these facial details are accurately captured in the shape space. Following the 3DMM literature [8, 14], we do not standardize the data for PCA, as all meshes are already aligned, oriented, and share the same units. Experimentally, we do not find the standardization to improve the optimization landscape either.

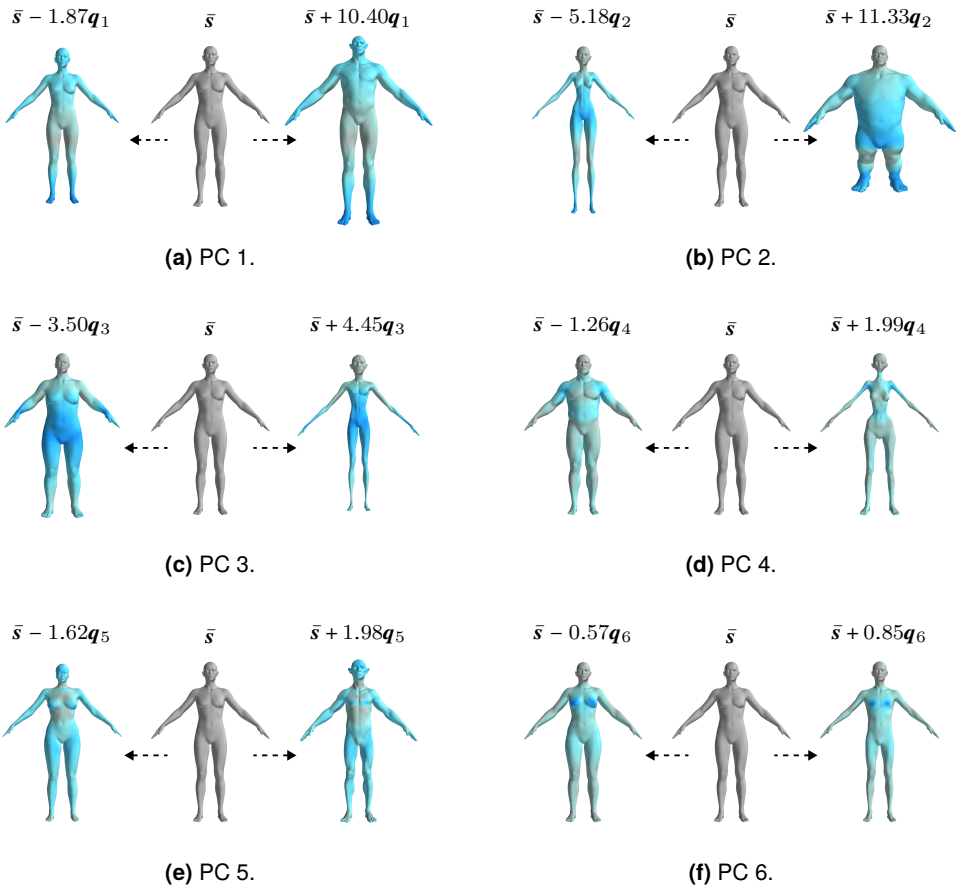
4.2.2 Axis-Aligned Bounding Box Clipping

Furthermore, we can constrain the shape space to remain close to the convex hull of the original data, which is the distribution of realistic shapes. Projecting the original data onto the shape space – i.e., the subspace spanned by the chosen subset of PCs – transforms the data points into a new space where each axis corresponds to a PC. Then, we can extract an axis-aligned bounding box (AABB) as the minimum bounding box that contains all the projected data points in the PC subspace. The sides of this bounding box are aligned with the axes of the chosen PCs, and not (necessarily) with the original feature axes. This AABB represents the range of the projected data along the selected PCs, encapsulating the variation in the data as captured by those components. Notably, limiting the shape space to the AABB of the training data does not impede extrapolation of the training data points, which would severely hinder the expressivity of the 3DMM. Intuitively, extrapolation of the shape space occurs outside the convex hull that is formed by projecting the training dataset to the PC shape space. Not only is the AABB a superset of the convex hull (the convex hull is *inscribed* in the AABB), but the volume spanned by the AABB is several orders of magnitude larger than that of the convex hull. We discuss this finding in more detail in Appendix 8.1.
With the 3DMM from Eq. (4.8), we can finally build our geometric prior $\mathbf{g}_{\boldsymbol{\phi}}(z)$ by substituting the weights ω_i with an activation function $\boldsymbol{\sigma}_{\boldsymbol{\phi}}(z)$ over the parameter vector z as:

$$\mathbf{g}_{\boldsymbol{\phi}}(z) = \bar{s} + \sum_{i=1}^{K} \sigma_{\boldsymbol{\phi}}(z_i)\boldsymbol{q}_i. \tag{4.9}$$

The activation function $\boldsymbol{\sigma}_{\boldsymbol{\phi}}(z) : \mathcal{Z} \to \mathcal{Z}'$ maps the parameter vector z to the axis-aligned bounding box

$$\mathcal{Z}' := [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_K, b_K]$$



(a) PC 1. **(b)** PC 2. **(c)** PC 3. **(d)** PC 4. **(e)** PC 5. **(f)** PC 6.

Figure 4.4: The first six body shape PCs, driven to the extremes of the AABB. The template mesh $\bar{s}$ is depicted at the center of each subplot. To the left and right, the meshes $\bar{s} + a_i q_i$ (minimum of the AABB) and $\bar{s} + b_i q_i$ (maximum of the AABB) are shown respectively for $i = 1, 2, \ldots, 6$. The labels above have substituted a_i and b_i with their corresponding actual scalar values. We illustrate the magnitude of the per-vertex offset in the unscaled PCs as a gradient of blue, where the intensity of the color correlates with the absolute displacement.

in K dimensions. The AABB is therefore the Cartesian product (denoted by "×") of closed intervals $[a_i, b_i]$, with a_i being the lower bound and b_i the upper bound of the i-th dimension, i.e.

$$a_i \leq b_i \forall i \in \{1, 2, \ldots, K\}\,.$$

We perform the clipping operation with the activation function $\sigma_{\boldsymbol{\phi}}(z)$ as:

$$\sigma_{\boldsymbol{\phi}}(z) = \boldsymbol{a} + (\boldsymbol{b} - \boldsymbol{a})\sigma_{\text{sigmoid}}(z), \tag{4.10}$$

where the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ simply aggregate the bounds of the AABB:

$$\boldsymbol{a} = (a_1, a_2, \ldots, a_K) \in \mathbb{R}^K, \quad \boldsymbol{b} = (b_1, b_2, \ldots, b_K) \in \mathbb{R}^K.$$

With a slight abuse of notation, the vector sigmoid function is determined by

$$\boldsymbol{\sigma}_{\text{sigmoid}}(\boldsymbol{z}) = \frac{1}{1 + e^{-\boldsymbol{z}}},$$

with the division and the exponential operators applied element-wise. We initialize the parameter vector $\boldsymbol{z}$ as $\boldsymbol{z}_{\text{init}}$ such that $\boldsymbol{\sigma}_{\boldsymbol{\phi}}(\boldsymbol{z}_{\text{init}}) = \mathbf{0}$ by inverting Eq. (4.10):

$$\boldsymbol{z}_{\text{init}} = \log\left(\frac{\boldsymbol{y}}{1 - \boldsymbol{y}}\right), \quad \text{with} \quad \boldsymbol{y} = -\frac{\boldsymbol{a}}{\boldsymbol{b} - \boldsymbol{a}}.$$

Again, we adapt the notation here to apply all operators element-wise and denote with $\boldsymbol{y}$ a temporary variable that we use just once to enhance readability. All in all, the generator $\mathrm{g}_{\boldsymbol{\phi}}(z)$ is fully parameterized by the tuple

$$\boldsymbol{\phi} := (\bar{\boldsymbol{s}}, \boldsymbol{q}, \boldsymbol{a}, \boldsymbol{b})\,.$$

4.2.3 Dataset

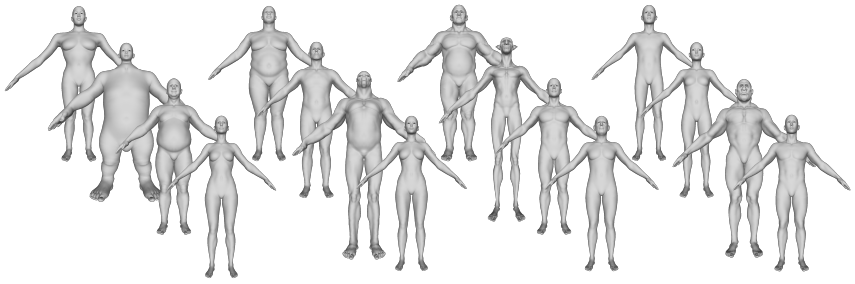


Figure 4.5: Some example meshes from our mesh dataset.

We gather a dataset of $|\mathbf{D}| = 236$ meshes representing a distribution of adult humanoid bodies. The dataset captures a wide range of anatomies and artistic styles that include realistic, fantastic, and cartoonish features. All samples in the dataset have been scuplted by professional technical artists holding to the highest quality standards of the AAA industry, as described in Chapter 3. Some example meshes are depicted in Fig. 4.5. Each mesh in the dataset consists of exactly $N = 25,182$ vertices and $M = 50,312$ quadrangular faces. While the positions of the vertices change across samples, the topology is the same for all. Thus, the meshes are considered to be blend shapes, meaning that any subset of the dataset can be linearly interpolated by blending the vertices of the selected meshes. In terms of [14], all our meshes have a point-to-point correspondance and share scale and orientation.

Furthermore, all meshes are registered to a common UV space that consists of

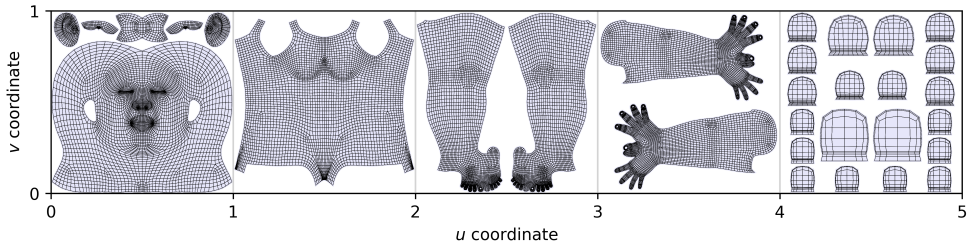


Figure 4.6: Mesh topology unwrapped into UV space, where each unit box represents an individual UDIM tile. Edges in the wireframe are drawn as black lines, and the inside of the faces is filled in light blue. From left to right, the UDIM tiles correspond to the head, body, legs, arms, and nails of the humanoid mesh.

five UDIM tiles for high resolution texturing, consisting of UV coordinates and vertex texture coordinate indices for efficient unwrapping and smooth blending along seams. In Fig. 4.6, we show the quadrangular topology of the dataset, unwrapped into UV space. Edge loops are optimized for rendering efficiency, as well as shape deformation (animation), subdivision and editability.

Besides, other vertex attributes are also shared across all meshes in the dataset.

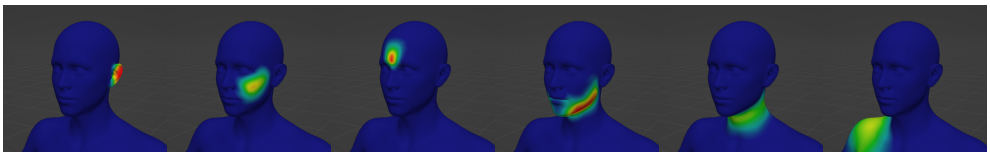


Figure 4.7: We render a selection of skinning weights of our meshes in Blender [12], which are displayed as a heat map (blue for a value of 0 and red for value of 1). From left to right, the weights correspond to the left ear, left cheek, center brow, lower jaw, neck, and right shoulder.

Our dataset also defines artist-designed skinning weights, which control the influence of a skeletal bone structure to each vertex. Skinning weights are critical for correct deformation of the surface in animation settings and must be designed jointly with the topology. We show exemplarily a selection of joints around the face and the upper body in Fig. 4.7. Each vertex has a weight for each joint in the skeleton, and weights are normalized to add up to 1 for each vertex. The underlying skeleton is composed of a hierarchy of over 200 bones, which we do not consider for our 3DMM. Our skinning weights are optimized to work with dual quaternion blend skinning [31].

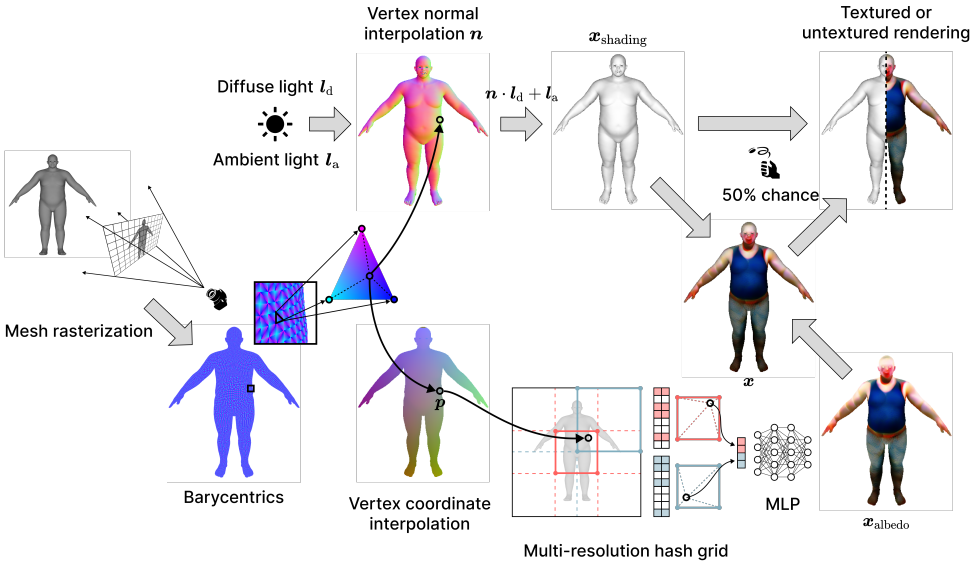


Figure 4.8: Demonstration of our differentiable foreground renderer, the background part is excluded of this illustration. First, we rasterize the input mesh with a differentiable rasterizer [37]. We use the rasterized barycentric coordinates of the triangles to interpolate different vertex attributes along the image plane. Two streams originate from here: one for albedo rendering (bottom), and another one for textureless Lambertian shading (top). For the albedo rendering, we interpolate the 3D vertex coordinates. Then, we pass the 3D position $\boldsymbol{p}$ of each foreground pixel through a multi-resolution hash encoding module [51] to extract a positional feature encoding. Our depiction of the multi-resolution hash encoding is inspired by InstantNGP [52]. The feature vector is decoded into an albedo color image via an MLP. On the upper shading stream, we interpolate the vertex normals to perform Lambertian shading with a random diffuse point light source and ambient light. During training, the final rendering consists of the shading and is combined with the albedo with a 50% chance.

4.3 Differentiable Renderer

In our pursuit of a forward model that generates an image $\mathbf{x}$ from a latent code $\mathbf{z}$, it is crucial that the forward model remains differentiable. This requirement is necessary to ensure that the latent code $\mathbf{z}$ can be optimized via backpropagation using gradient-based methods. In the previous section, we discuss the first component of this forward model: a generator $\mathbf{g}_{\boldsymbol{\theta}}$ that translates the latent code $\mathbf{z}$ into a 3D mesh. Having established this foundation, we now turn our attention to the second essential component, which is the differentiable renderer

$$\mathbf{r}_{\boldsymbol{\theta}} : \mathcal{M} := \mathbb{R}^{3N} \rightarrow [0,1]^{D}, \tag{4.11}$$

which rasterizes the 3D mesh $\mathbf{g}_{\boldsymbol{\phi}}(z)$ into the image $\mathbf{x}$ given the camera parameters $\mathbf{c}$:

$$\boldsymbol{x} = \mathbf{r}_{\boldsymbol{\theta}}\big(\mathbf{g}_{\boldsymbol{\phi}}(z), \boldsymbol{c}\big). \tag{4.12}$$

The geometric prior defines a shape space that is regularized by a 3DMM, but it does not represent texture color. Thus, we build our differentiable renderer not only to rasterize the mesh from our mesh generator, but also to learn a feature space of RGB albedo color and apply shading effects. A schematic representation of our renderer can be found in Fig. 4.8.
Given a mesh $\mathbf{g}_{\boldsymbol{\phi}}(z)$ into the image $\mathbf{x}$ and camera parameters $\mathbf{c}$, we first normalize the input mesh to the unit cube. When rendering a 3D scene, we need to account for the perspective distortion that occurs when projecting 3D objects onto a 2D screen. We use nvdiffrast [37] to project the barycentric coordinates onto the 2D image plane with perspective correction and occlusions. Nvdiffrast is a GPU-based differentiable rasterization library used in deep learning frameworks for rendering 3D models and computing gradients for use in training. The nvdiffrast API expects input vertices to be in clip coordinates. In this context, nvdiffrast applies the barycentric perspective correction during the rasterization process to ensure that the interpolation of vertex attributes (like textures) is accurate when viewed in perspective.

Albedo. The projected barycentric coordinates can then be used to interpolate vertex attributes, such as vertex normals and 3D positions across the surface of a projected triangle. For albedo texture, we create an image $\boldsymbol{x}_{\mathrm{pos}}$ with interpolated 3D vertex positions and sample for each pixel the point color at that 3D coordinate from an InstantNGP module [52] that leverages the efficient implementation in tinycudann [51]. The 3D position is encoded into a feature vector via a multi-resolution hash grid, which consists of a series of voxel grids at different resolution levels. For that purpose, we first find the corners of the surrounding voxels at each resolution level, which are all assigned an integer index, and look up a feature vector corresponding to each corner in a hash grid. The features are then interpolated according to the relative position of the 3D coordinate within each voxel. The interpolated features are concatenated together and taken as input to an MLP that has three output neurons, one for each color channel, and a sigmoid activation to limit outputs to the range $(0, 1)$. The output is an albedo image $\boldsymbol{x}_{\mathrm{albedo}}$. All components are differentiable and learnable, InstantNGP learns neural field of albedo color. Intuitively, our renderer can be thought of as a cloud of RGB color (the InstantNGP) that is only painted onto the surface of a mesh canvas (the mesh $\mathbf{g}_{\boldsymbol{\phi}}(z)$). The gradients provide a signal to deform the 3D canvas and correct the 3D color cloud to better represent the shape and appearance predicted by the difusion prior in 2D image space.

Shading. We consider a Lambertian shading model [38] to provide an additional learning signal that proves to be critical to learn proper shapes. Without shading, our differentiable renderer produces flat projections of the albedo texture and our generative framework can "cheat" by painting geometric detail on surfaces that have a different shape. We interpolate vertex normals along the rasterized image and obtain an image $\boldsymbol{x}_{\text{normal}}$. Following DreamFusion [56], we consider a random point light source that has the position $\boldsymbol{l}$ and color $\boldsymbol{l}_{\text{p}}$, as well as an ambient light source with color $\boldsymbol{l}_{\text{a}}$. We assume $\boldsymbol{l}$, $\boldsymbol{l}_{\text{p}}$, and $\boldsymbol{l}_{\text{a}}$ to be repeated for each pixel to have the same length as $\boldsymbol{x}_{\text{pos}}$. We can then obtain the directions $\boldsymbol{l}_{\text{d}}$ of the point light source in image space as the difference

$$\boldsymbol{l}_{\text{d}} = \boldsymbol{l} - \boldsymbol{x}_{\text{normal}}.$$

Then, the shading image $\boldsymbol{x}_{\text{shading}}$ is computed as:

$$\boldsymbol{x}_{\text{shading}} = \boldsymbol{x}_{\text{normal}} \cdot (\boldsymbol{l}_{\text{d}} \odot \boldsymbol{l}_{\text{p}}) + \boldsymbol{l}_{\text{a}},$$

where $\odot$ denotes the point-wise Haddamard product. With a probability of 50%, we use the textureless $\boldsymbol{x}_{\text{shading}}$ as the output of differentiable renderer, and the remaining times we return the fully shaded image $\boldsymbol{x}$ as:

$$\boldsymbol{x} = \boldsymbol{x}_{\text{shading}} \odot \boldsymbol{x}_{\text{albedo}}.$$

Background. For the background image, we follow MVDream [67] to create a neural environment background map. We compute viewing directions $\boldsymbol{x}_{\text{view}}$ as

$$\boldsymbol{x}_{\text{view}} = \boldsymbol{x}_{\text{pos}} - \boldsymbol{c},$$

where $\boldsymbol{x}_{\text{pos}}$ designates the image of interpolated 3D vertex positions and $\boldsymbol{c}$ are the camera parameters. Next, we encode $\boldsymbol{x}_{\text{view}}$ using spherical harmonics. We then predict a per-pixel RGB color with an MLP that shares the same architecture as the the feature MLP for albedo rendering. To achieve sparsity between the foreground and the background, we replace the background with a probability of 50% with a random, uniform color.

4.4 Diffusion Prior

At inference, we need to find the parameter set z and corresponding multi-resolution hash grid parameters that minimize the loss under some critic network. As previously discussed, we adopt SDS to provide guidance to the optimization process with a diffusion model. In this section, we first examine the choice of a low CFG scale (in comparison with other SDS methods), enabled by the information flow that exists between the geometric prior and the denoising diffusion network. Then, we motivate using MVDream [67] as our pre-trained diffusion prior, due to its architecture that supports supervision on multiple views of the 3D object.

4.4.1 Classifer-Free Guidance in Score Distillation Sampling

DreamFusion [56] originally proposes to set a CFG scale of $w_{\text{CFG}} = 100$, which is orders of magnitude larger than image sampling methods. The authors argue that the large value of w_{CFG} is necessary for the SDS objective to converge to a mode: an increasing CFG scale aligns sampled images more closely with the conditioning prompt, while decreasing the diversity of those samples. Subsequent SDS-based 3D generative methods also rely on comparably high CFG weights [41, 67]. However, large CFG scales also cause $\boldsymbol{x}_{0,\text{CFG}}$ (we denote with $\boldsymbol{x}_{0,\text{CFG}}$ the reconstructed image with CFG) to have a large scale and, by extension, the diffused image to be overexposed [1]. Therefore, the guidance weight w_{CFG} practically poses a trade-off between generating blurry 3D models for low values and over-saturated results for high values.
Inspired by Alldieck et al. [1], we decompose $\nabla_z \mathcal{L}_{\text{SDS}}$, presented in Eq. (3.11), into two components:

$$\nabla_z \mathcal{L}_{\text{SDS}} = \nabla_z \mathcal{L}_{\text{proj}} + (w_{\text{CFG}} + 1)\, \nabla_z \mathcal{L}_{\text{cond}} \tag{4.13}$$

$$\nabla_z \mathcal{L}_{\text{proj}} = \mathbb{E}_{t,\boldsymbol{\varepsilon}} \left[w(t) \left(\boldsymbol{\varepsilon}_{\boldsymbol{\phi}}(\boldsymbol{x}_t; t) - \boldsymbol{\varepsilon} \right) \frac{\partial \boldsymbol{x}_t}{\partial z} \right] \tag{4.13a}$$

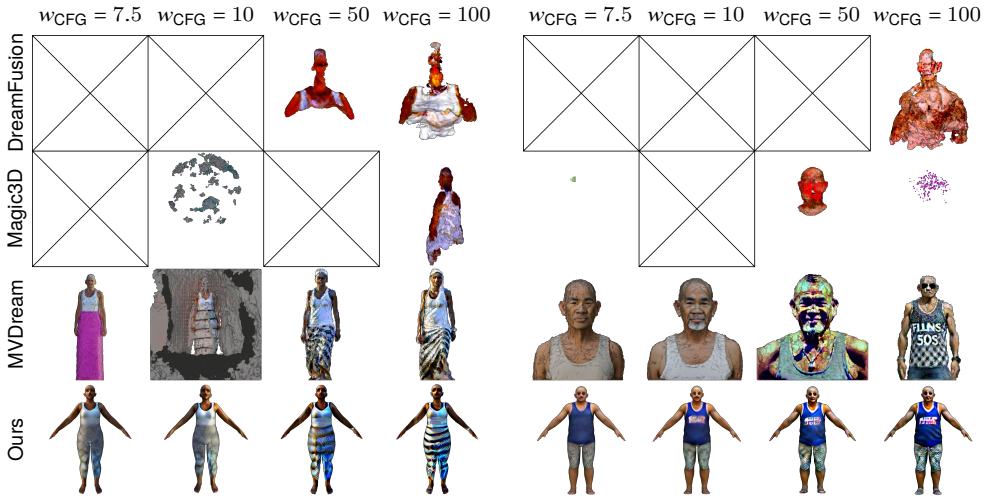
$$\nabla_z \mathcal{L}_{\text{cond}} = \mathbb{E}_{t,\boldsymbol{\varepsilon}} \left[w(t) \left(\boldsymbol{\varepsilon}_{\boldsymbol{\phi}}(\boldsymbol{x}_t; \boldsymbol{y}, t) - \boldsymbol{\varepsilon}_{\boldsymbol{\phi}}(\boldsymbol{x}_t; t) \right) \frac{\partial \boldsymbol{x}_t}{\partial z} \right] \tag{4.13b}$$

Please, refer to Appendix 8.2 for the derivation of Eq. (4.13). Note that our decomposition is not identical to [1], as we consider the original formulation of w_{CFG} by [23] and also factor $(w_{\text{CFG}} + 1)$ out of $\nabla_z \mathcal{L}_{\text{cond}}$. Nevertheless, the subsequent mathematical analysis is fundamentally equivalent, as it is rooted in the fact that there is an unconditional term (4.13a) and a conditional term (4.13b) that contribute to the SDS gradient. On the one side, $\nabla_z \mathcal{L}_{\text{proj}}$ guides the unconditional reverse diffusion process by contributing the gradient to remove one level of noise from the perturbed image $\boldsymbol{x}_t$ with noise scale t. The gradient points towards the manifold of realistic images (or the distribution the diffusion model has been trained on). On the other side, $\nabla_z \mathcal{L}_{\text{cond}}$ directs the conditional reverse diffusion process to align with the provided prompt by supplying the gradient to the image denoised via conditioning on the text description $\boldsymbol{y}$. Particularly, the latter term does not necessarily point towards the manifold of realistic images, which further explains the over-saturation artifacts observed in 3D models sampled with high CFG scales, where $\nabla_z \mathcal{L}_{\text{cond}}$ dominates the SDS objective (4.13).
We argue that the vanilla SDS objective is underconstrained, which explains the need for an exaggerated guidance scale to aid mode seeking. We can find a mode using the same CFG scale as image sampling methods by leveraging a geometric prior that constrains the space of admissible shapes to a class-specific manifold that intersects the distribution of realistic shapes. In turn, we empirically find generated 3D models to improve both in shape and appearance when lowering the

guidance scale (see Fig. 4.9), because we can sample the diffusion prior with optimal hyperparameter settings. There is an information flow from the geometric prior $\mathbf{g}_{\boldsymbol{\phi}}(z)$ to the diffusion prior $\nabla_z \mathcal{L}_{\text{SDS}}$ via $\boldsymbol{x}_t$ via the forward model. By inserting the equation for the forward model in Eq. (4.12) into the forward diffusion process in Eq. (3.4) and setting $\boldsymbol{x}_0 := \boldsymbol{x}$, we obtain:

$$\boldsymbol{x}_t = \mathcal{N}\big(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{r}_{\boldsymbol{\theta}}\big(\mathbf{g}_{\boldsymbol{\phi}}(z), \boldsymbol{c}\big), (1-\bar{\alpha}_t)\,\mathbf{I}\big). \tag{4.14}$$

Thus, per Eq. (4.14), the denoising U-Net $\boldsymbol{\varepsilon}_{\boldsymbol{\phi}}(\boldsymbol{x}_t; t)$ that predicts the noise added to $\boldsymbol{x}_t$ always operates on perturbed, textured projections $\boldsymbol{x}_t$ of the parameter vector z, which is a realization of the shape space $\mathcal{Z}$ of the geometric prior. Therefore, both the unconditional gradient $\nabla_z \mathcal{L}_{\text{proj}}$ and the text-conditional gradient $\nabla_z \mathcal{L}_{\text{cond}}$ are heavily biased to point to images that are a possible projections of the shape space $\mathcal{Z}$ of the 3DMM. The geometric prior acts as prior of the shape space and it regularizes the diffusion model via Eq. (4.14). We provide the intuition that this effect constrains the SDS objective by guiding it towards the manifold that intersects the distribution of realistic images with the shape space of the geometric prior (which is our AABB of the data in PCA space), alleviating the need for a strong contribution of $\nabla_z \mathcal{L}_{\text{cond}}$ for mode-seeking via a high value of w_{CFG}. A similar reasoning could be followed to argue that we can converge to valid shapes, even without a multi-view consistent diffusion prior (see experiment in Sec. 5.6), although we find multi-view consistency to improve convergence. In Fig. 4.9, we ablate the effect of the guidance coefficient w_{CFG} on the meshes sampled by our and comparable methods, such as DreamFusion [56], Magic3D [41], and MVDream [67]. For DreamFusion and Magic3D, we use the implementation in threestudio [18] with StableDiffusion v2.1 [58]. For MVDream, we employ the official implementation that is also available through threestudio. In the case of DreamFusion, we observe that the algorithm reliably produces outputs only at the prescribed value of $w_{\text{CFG}} = 100$. We find Magic3D to be more unstable, where no single value consistently yields optimal results, though values exceeding $w_{\text{CFG}} = 50$ are generally required. MVDream on the other side exhibits greater stability, with most w_{CFG} values producing perceptually satisfactory results. Nevertheless, we see that lower w_{CFG} values can still introduce artifacts in certain scenarios, which aligns with the authors' recommendation of using $w_{\text{CFG}} = 50$. Across all three concurrent methods, cloud-like artifacts such as structural holes, extrusions, and floating blobs of material are evident, which are known issues of neural field representations like NeRF [50] and InstantNGP [52]. Our method, in contrast, is the most stable and reliable of all, where the w_{CFG} parameter primarily influences the saturation of colors in the albedo texture and 3D shape to a much lesser degree. While $w_{\text{CFG}} = 7.5$ is generally considered to work well for image-based StableDiffusion applications, we find the slightly higher value of $w_{\text{CFG}} = 10$ to enhance details in our use case.



(a) "a middle-aged female bedouin dressed in white"

(b) "filipino male, in his 50s, looks younger"

Figure 4.9: Effect of the guidance weight w_{CFG} on DreamFusion [56] (first row), Magic3D [41] (second row), MVDream [67] (third row) and our method (fourth row). We render an orthographic frontal projection of the mesh output and place a black cross wherever the isosurface extraction algorithm fails to extract a mesh from the volume. While colors saturate with increasing value of w_{CFG}, most SDS methods require high guidance scale: DreamFusion recommends $w_{\text{CFG}} = 100$, and Magic3D and MVDream use $w_{\text{CFG}} = 50$. We find our approach to produce best results with $w_{\text{CFG}} = 10$. All meshes are generated by appending the string ", shaved head, wearing a tank top" to the labels in the captions.

4.4.2 Multi-view Consistency

As a diffusion prior, we opt for the diffusion model provided by MVDream [67], which is trained on multi-view renderings of Objaverse [13] objects and can predict orthogonal views of a 3D asset that have an acceptable degree of multi-view consistency. While the 3D consistency of the diffused images is not perfect, we find MVDream to clearly outperform Zero123 [43], Zero123-XL [13], and SyncDreamer [44], which all address the same task of multi-view image diffusion.
MVDream finetunes Stable Diffusion [58] by making a key architectural change to support multi-view images. The MVDream network incorporates a pseudo-3D self-attention mechanism that simply flattens the concatenated input views along the sequence dimension representing the tokens. Hence, the 2D self-attention layer from the original Stable Diffusion U-Net can be recycled by expanding it to support the additional input weights. The extended self-attention layer is initialized by repeating the pre-trained weights for the additional input views, and the weights are later finetuned and provided in the checkpoint. Finally, the finetuned U-Net is also conditioned on camera embeddings by encoding the camera parameters with a

2-layer MLP. The camera embeddings are then added to the time embeddings as residuals.
We also adopt the reconstruction objective from MVDream, which is a special case of the SDS loss:

$$\mathcal{L} = \mathbb{E}_{t,\boldsymbol{c},\boldsymbol{\varepsilon}} \left[\|\boldsymbol{x} - \hat{\boldsymbol{x}}_0\|_2^2 \right], \tag{4.15}$$

where $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is isotropic Gaussian noise, $\boldsymbol{c}$ is a randomly sampled camera, and t is the diffusion time step (we refer to the implementation details in Sec. 5.1). MVDream demonstrates that Eq. (4.15) is equivalent to the original SDS formulation from Eq. (3.11) by setting the weights $w(t)$ equal to the signal-to-noise-ratio. The modified loss performs similarly to the SDS objective. The advantage of having a reconstruction loss like Eq. (4.15) is that the CFG rescaling trick from [42] can be applied to improve the noise schedule of the diffusion model.

5 Experiments

In this section, we explore potential applications of our method and provide results. We start by explaining the experimental setup and implementation details in Sec. 5.1. We then propose an evaluation metric in Sec. 5.2 that interpolates the image embeddings of renderings from different views to measure the alignment between the sampled 3D mesh and the input text prompt. Next, in Sec. 5.3, we evaluate the pipeline that we propose for text-to-3D generation. Then, we show in Sec. 5.4 how output meshes can be edited by manually adjusting the coefficients that scale the contributions of the PCs, or by exchanging textures in a shared UV space. Sec. 5.5 demonstrates that our generative framework is compatible with other 3DMMs as a geometric prior, which allows to extend the range of our generative model to other asset classes. Similarly, we interchange the diffusion prior in Sec. 5.6 to modify the guidance of our method, which can be useful, for instance, to generate meshes in a particular style. Finally, we ablate some hyperparameter choices in Sec. 5.7.

5.1 Experimental Setup and Implementation Details

We manually curate a test set of prompts that are in the distribution of the geometric prior. Also, we add a procedural prompt processing step that mitigates artifacts that arise when the diffusion prior guides the generation process towards shapes outside the shape space of the 3DMM. For example, the geometric prior that we construct defines a shape space that captures variation in the human anatomy, but it does not represent items like garments or hair. Hence, in that case we append ", shaved head, wearing a tank top" to every prompt to ensure that the diffusion prior only considers clothing items and hair that remain in tight proximity of the body of the character, and can therefore be represented exclusively in texture space without additional geometry in our 3D representation. Similar prompt engineering tricks are applied for geometric priors that span different shape spaces (such as busts of human faces), which we indicate in the corresponding section (Sec. 5.4). For the negative prompt, we keep the default string from MVDream [67] which is "ugly, bad anatomy, blurry, pixelated obscure, unnatural colors, poor lighting, dull, and unclear, cropped, lowres, low quality, artifacts, duplicate, morbid, mutilated, poorly drawn face, deformed, dehydrated, bad proportions".
We synthesize new meshes by running the optimization process for 7000 iterations.

We operate MVDream on 4 views at 64×64 resolution for the first 500 iterations, and then at 256×256 for the remaining iterations. All views share the same field of view, which we sample from a uniform distribution $\mathcal{U}(15°, 60°)$ and the elevation from $\mathcal{U}(0°, 30°)$. The camera distance is derived by multiplying the object size with the NDC focal length and a factor sampled uniformly from $\mathcal{U}(1.0, 1.2)$. The azimuth of the first view is sampled from $\mathcal{U}(-180°, 180°)$ and the remaining views are uniformly distributed to cover the entire range with $90°$ offsets. We keep the batch size at 4 and enable soft shading from the first iteration.
The multi-resolution hash grid has 16 levels with each 2 features and a base resolution of 16. The feature MLPs for albedo color and background color consist of a single hidden layer of width 64 with ReLU activation and a 3-channel (RGB) output layer with sigmoid activation. We use the AdamW optimizer [34, 47] with the coefficients $\beta_1 = 0.9$, $\beta_2 = 0.99$, the term $\varepsilon = 1 \times 10^{-15}$ for numerical stability, and weight decay $\lambda = 0.01$. We set the learning rate for the shape parameters z and the multi-resolution hash grid encoding to $\gamma_z = \gamma_{\text{encoding}} = 0.01$, and the learning rate of the MLP albedo network and the background to $\gamma_{\text{albedo}} = \gamma_{\text{background}} = 0.001$. All remaining hyperparameters are kept to the default in MVDream. The entire SDS optimization takes around 20 minutes on an Nvidia Tesla A100 GPU.

5.2 Multi-view Contrastive Language-Image Pre-Training Similarity Score

Evaluating the quality of 3D generative models is a challenging task, primarily because the most commonly used metrics – Fréchet Inception Distance (FID) [21], Inception Score (IS) [61], Contrastive Language-Image Pre-Training (CLIP) Score [57], and CLIP R-Precision [54] – are designed for assessing 2D images rather than 3D objects. FID and IS are effective at measuring the quality and diversity of generated images but fall short when evaluating how well a 3D mesh adheres to its intended structure. These metrics operate solely on 2D projections, ignoring the inherent 3D nature of the objects they represent. The CLIP score and CLIP R-Precision, on the other hand, assess the alignment between a generated image and a textual prompt, yet both metrics also fail to account for the 3D coherence of the object across different views. None of these evaluation criteria adequately capture the full spatial and structural consistency that is critical when evaluating 3D meshes.
There are also metrics that measure the shape dissimilarity between point clouds, such as the Chamfer Distance. However, the Chamfer Distance requires a dataset of diverse ground truth captioned meshes, which we do not have for our generative task. Therefore, we extend the standard CLIP score to better assess 3D meshes by considering multiple views of an object. The multi-view CLIP score (MV-CLIP) metric interpolates the embeddings from different views of the 3D object, providing

a more holistic measure of how well the object adheres to its corresponding textual prompt across various perspectives.
We introduce a new notation only for this section to avoid cluttering the variable names with too many sub-labels. Let $\boldsymbol{y}$ represent a textual prompt and $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$ denote a set of N images corresponding to different views of a 3D mesh. For each view $\boldsymbol{x}_i$, we obtain the corresponding image embedding $\mathbf{v}_i$ using a pre-trained CLIP model:

$$\mathbf{v}_i = \text{CLIP}_{\text{image}}(\boldsymbol{x}_i), \tag{5.1}$$

where $\mathbf{v}_i \in \mathbb{R}^{B\times 512}$ with batch size B. Then, we compute the text embedding $\mathbf{u}$ for the prompt $\boldsymbol{y}$ using the same CLIP model

$$\mathbf{u} = \text{CLIP}_{\text{text}}(\boldsymbol{y}), \tag{5.2}$$

and interpolate the image embeddings by averaging the unnormalized embeddings

$$\mathbf{v}_{\text{interpolated}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{v}_i. \tag{5.3}$$

Next, we normalize the interpolated image embedding and the text embedding:

$$\hat{\mathbf{v}}_{\text{interpolated}} = \frac{\mathbf{v}_{\text{interpolated}}}{\|\mathbf{v}_{\text{interpolated}}\|} \quad \text{and} \quad \hat{\mathbf{u}} = \frac{\mathbf{u}}{\|\mathbf{u}\|}. \tag{5.4}$$

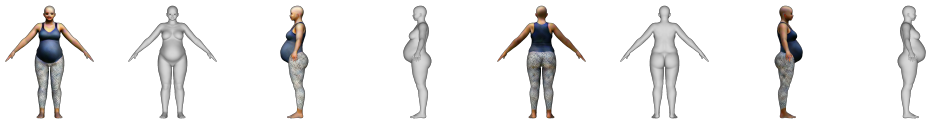
We compute the cosine similarity between the normalized embeddings and apply a learned scaling factor σ from the CLIP model:

$$\text{MV-CLIP} = \sigma \cdot \hat{\mathbf{v}}_{\text{interpolated}}^{\top}\hat{\mathbf{u}}. \tag{5.5}$$

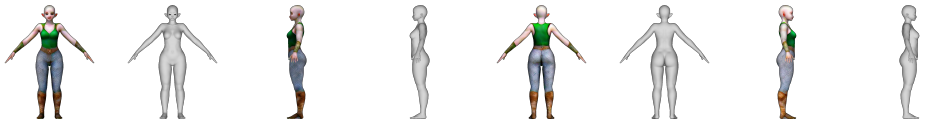
Furthermore, we can also derive a MV-CLIP win ratio, which compares a set of generative models (baselines) by sampling one mesh from each model and counting the fraction of times where MV-CLIP retrieves the mesh that corresponds to each model (the *winner* model) given the text prompt. The MV-CLIP win ratio can thus be interpreted as a precision score of a MV-CLIP retrieval system, which operates on a set of distractors – the outputs of each baseline – that compete with each other to maximize their likelihood of being selected.
It is important to note that while the proposed metrics offer a more comprehensive assessment than existing 2D metrics, it remains a proxy metric for text-to-mesh alignment and does not capture the quality of the 3D model. The MV-CLIP metric primarily evaluates how well the model's visual outputs align with the prompt across different views, but it does not measure the true 3D coherence or physical accuracy of the generated mesh.

5.3 Text-to-3D Asset Generation

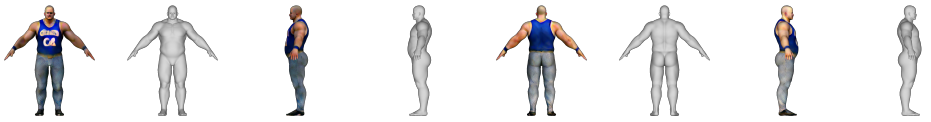
The core application of our method is text-to-3D asset generation. In this section, we show a selection of sampled meshes and benchmark our method with other state-of-the-art text-to-3D models both qualitatively and quantitatively.



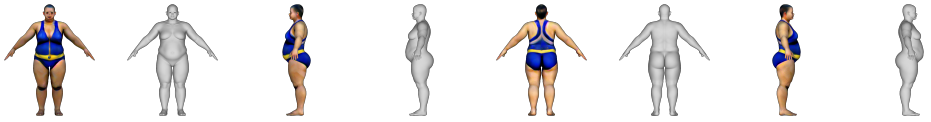
(a) “9-month pregnant woman of color”



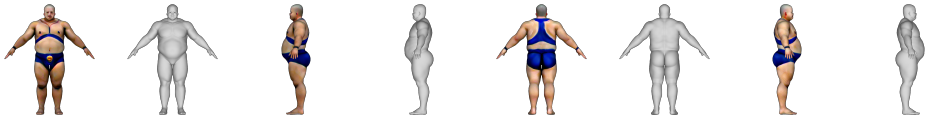
(b) “An elf princess with pointy ears and long fingernails”



(c) “John Cena with a beer belly”



(d) “Female sumo wrestler”



(e) “Male sumo wrestler”

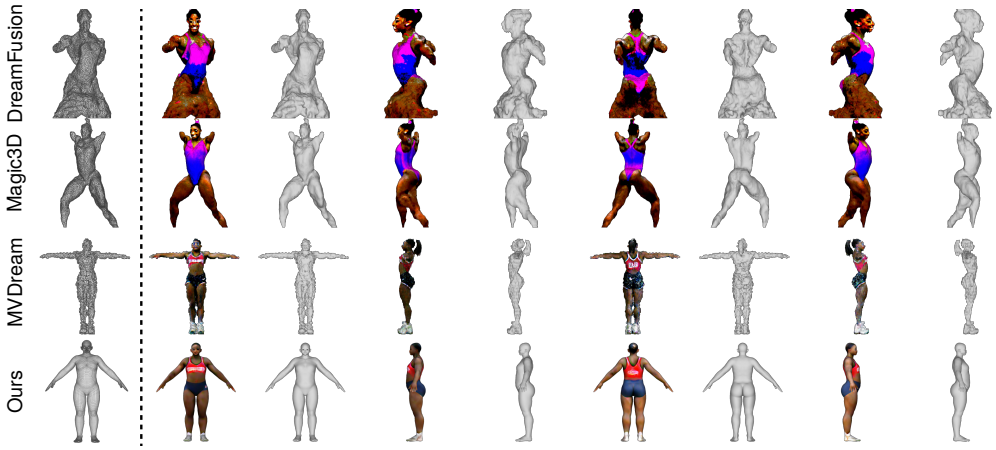
Figure 5.1: Front, left, back, and right orthographic views of meshes synthesized with our method. Each view is rendered in MeshLab [11] with and without albedo texture to capture the geometric detail of the shape surface. The meshes are generated by appending the string “, shaved head, wearing a tank top” to the labels in the captions.

Qualitative analysis. In Fig. 5.1, we display a set of of sampled meshes across four different views that cover the entire azimuth range uniformly. The examples cover a wide variety of genders and body shapes. We render the colored mesh with albedo map and the textureless mesh with shading effect. The untextured image allows to better recognize the structural detail of the surface. Otherwise, potential artifacts like unsmooth surfaces, holes, extrusions, or fused body parts could be

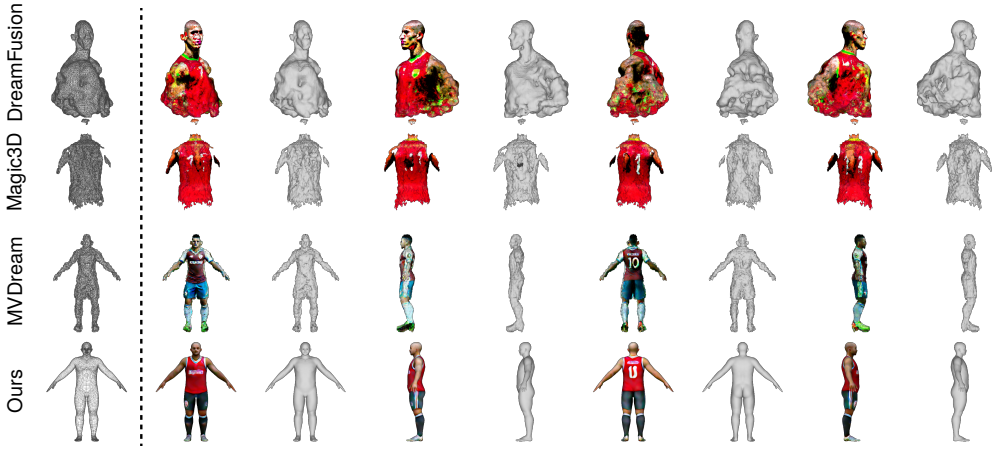
hidden by the albedo texture. More results can be found in Fig. 8.2 in Appendix 8.3, which extends the selection of samples that we present here.
Performing a qualitative analysis of the output meshes exhibited here, we see a rich variety of output shapes and textures. The synthesized meshes are coherent with the provided prompt and can be used directly in industrial applications without requiring post-processing. The quadrangular topology of the outputs holds up to high quality standards for realistic surface deformation and efficient rendering. Moreover, the anatomy of the body shapes is highly plausible: all characters have five fingers in the hands and feet, the bodies are quasi-symmetrical along the sagittal plane (small-scale asymmetries that occur in the real world like nose bending, irregular head shape, or slightly differently long limbs are still supported), and the meshes are multi-view consistent and do not suffer from fused or self-intersecting elements. Interestingly, the anatomical accuracy of the 3D humanoids may even outperform that of 2D image diffusion models, which are known to produce uncanny faces and unrealistic hands and limbs [55]. To the best of our knowledge, the detail around facial features or muscles is unparalleled by any other text-to-3D generative model that relies on neural fields as a scene representation.

Qualitative benchmarks. We benchmark our method against existing baselines, such as DreamFusion [56], Magic3D [41], and MVDream [67]. We use the implementations available through threestudio [18], which is not the official implementation in the case of DreamFusion and Magic3D. There are some differences that we list here. On the one hand, DreamFusion officially uses Imagen [60] and Magic3D leverages eDiff-I [5], which are both diffusion models that are not publicly available. Instead, we use the open-source StableDiffusion v2.1 latent diffusion model [58] in both cases. Furthermore, we use InstantNGP [52] instead of mip-NeRF [7] as the scene representation for DreamFusion. For MVDream on the other hand, the implementation in threestudio is officially supported.
The selected baselines are particularly interesting for our evaluation setting, not only because of their state-of-the-art performance in text-to-3D mesh synthesis, but also because they share several technical details with our pipeline. DreamFusion introduces the concept of SDS, which lies at the core of our method. Magic3D also uses both multi-resolution hash encoding and mesh rasterization for the differentiable renderer. However, instead of directly combining both scene representations into a single differentiable renderer like we do, Magic3D applies two separate optimization stages instead. The first stage extracts a coarse mesh from the neural field, and the second part refines the mesh. Finally, the central contribution of MVDream is to finetune StableDiffusion with minimal modifications to the self-attention layers of the denoising U-Net [59] to enable multi-view image synthesis with enhanced 3D consistency. We use the multi-view diffusion network from MVDream for as our diffusion prior.

In Fig. 5.2, we observe that our method is the only of the four that outputs clean and



(a) “Simone Biles with a beer belly”



(b) “A young Peruvian soccer player with an athletic build”

Figure 5.2: We benchmark our method (fourth row) against DreamFusion [56] (first row), Magic3D [41] (second row), and MVDream [67] (third row). As seen in the first column, ours is the only approach capable of synthesizing meshes with clean topology. The remaining eight columns show front, left, back, and right orthographic views rendered in MeshLab [11] with and without albedo texture. Our method uniquely produces smooth surfaces and realistic body anatomy. All meshes are generated by appending the string “, shaved head, wearing a tank top” to the labels in the captions.

consistent topology. The high-polygon, irregular triangular topology of the meshes extracted from the volumetric representations of our baselines makes the meshes unsuitable for production use without extensive manual refinement. Besides rendering performance issues, the topology is inherently problematic for deformation, as the angular and irregular creases around edge loops lead to folds that disrupt

Model	MV-CLIP B/32 (↑)		MV-CLIP B/32 win ratio (↑)	
	Color	Shape	Color	Shape
DreamFusion	23.80	22.15	0	$\mathbf{0.\overline{3}}$
Magic3D	22.60	20.5	0	$\mathbf{0.\overline{3}}$
MVDream	25.22	21.84	$0.1\overline{6}$	0
Ours	$\mathbf{26.68}$	$\mathbf{22.42}$	$\mathbf{0.8\overline{3}}$	$\mathbf{0.\overline{3}}$

Table 5.1: We evaluate the multi-view alignment of generated meshes with their corresponing text prompt. We measure the MV-CLIP Score and the MV-CLIP win ratio on textured renderings (color) and textureless shading renderings (shape) respectively. Due to compute constraints, we only run one seed per prompt.

smooth movement and animation. Moreover, both DreamFusion and Magic3D suffer heavily from artifacts like the Janus problem and content drift, which is illustrated in the concrete example in Fig. 5.2b. First, for DreamFusion, we see that the the arms of the soccer player fade and merge into the shirt, where there is no clear separation between different body parts. There is no recognizable anatomical structure in the torso of the soccer player in the textureless shading rendering, which exposes the underlying shape of the mesh. Second, Magic3D produces only the shirt of the soccer player, but without the human that wears it. Even for the side views, the shirt appears to be rendered from a frontal or posterior view, where both sleeves are visible to the left and right. In general terms, MVDream meshes demonstrate much higher perceptual quality compared to the previous baselines and do not display apparent multi-view inconsistencies. Nevertheless, the surface is highly unsmooth, the texture looks over-saturated, mesh elements that are in close proximity to each other fuse together, and there is missing detail around areas like the fingers in the hands. A figure with additional prompts for comparison with the baselines can be found in Fig. 8.3 in Appendix 8.3. In the additional examples, DreamFusion and Magic3D fail to produce any meaningful output in certain cases, where either no mesh is extracted by the isosurface algorithm or the mesh consists of floating blobs of material that do not appear to display any recognizable pattern. All in all, we find our method to be most reliable and produce the most natural colors, realistic and anatomically accurate shapes, predictable geometry with artist-designed topology and UVs, and closest alignment with the input prompt.

Quantitative benchmarks. Our qualitative findings are further supported by a quantitative analysis in Tab. 5.1, where we measure the MV-CLIP metric and the MV-CLIP win rate introduced in Sec. 5.2. Both scores indicate a better match between the sampled mesh and the input prompt as the values increase. Due to compute restrictions, we calculate the scores on a test set of 6 prompts with fixed

seed and use the official CLIP ViT-B/32 checkpoint [57]. Following DreamFusion, we consider fully shaded renderings with albedo texture (color) and without texture (shape). The untextured renderings show the true shape of the synthesized mesh, which can otherwise be camouflaged by the albedo texture. Textureless images are likely out of the training distribution of the CLIP ViT-B/32 network, which suggests that the measure on the shape renderings could be more unpredictable. However, we still find the score on the shape to be an informative proxy metric.
Our method clearly outperforms all the baselines across all metrics and ties with DreamFusion and Magic3D in MV-CLIP win ratio for shape. The average MV-CLIP score is considerably higher for our method compared to the other ones, both for color and for shape. Out of the six examples in the training dataset, our method wins on the colored mesh retrieval task with MV-CLIP score five times, and MV-Dream wins the remaining one. For shape, DreamFusion, Magic3D, and ours each win one time.
Despite all the positive findings, we still want to highlight that our method has a unique disadvantage. Our generative framework is not capable of modeling garments, hair, or accessories in shape space. Furthermore, like all the baselines, our method also inherits the gender and racial bias from the pre-trained 2D diffusion prior. We find that most prompts tend to generate Caucasian males by default unless we specify it otherwise in the prompt.

5.4 Output Editing

Figure 5.3: We show the original output in (i). In (ii) we drive the PC 1 down, making the character shorter, and in (iii) we raise the coefficient of PC 2, which creates a wider figure. In (iv) and (v), we switch the textures of the original mesh in (i) with the textures from other generated models.

Our proposed framework has the property that shape and appearance are disentangled in the generation process. The shape is controlled by the parameter vector z that drives the 3DMM in the geometric prior (Sec. 4.2), while the appearance is determined by the parameters θ of the InstantNGP [52] at the core of the differentiable renderer (Sec. 4.3). Therefore, one more advantage of using the shape space of a 3DMM for mesh generation is that the output shapes are interpretable.

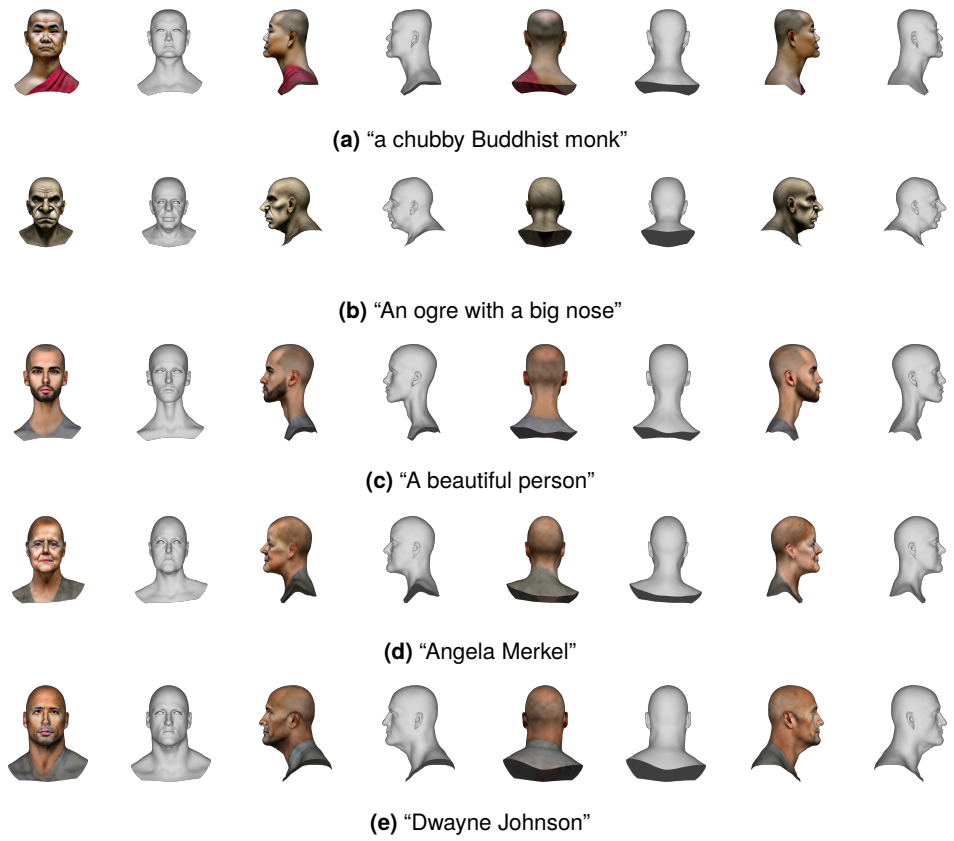
Specifically, we can investigate the coefficient vector $\boldsymbol{\sigma_\phi}(z)$ associated with a synthesized mesh to assess how much each PC contributes to the generated shape. This observation enables an additional manual editing workflow, where we can drive individual PCs up or down to modulate their impact on the output shape. Furthermore, all meshes are mapped to the same UV space, making it also uniquely possible to switch textures between generated assets.

Shape editing. For example, we take the pregnant woman in Fig. 5.1a and modify her shape. We store the vector $\boldsymbol{\sigma_\phi}(z)$ as additional output and make the observation that PC1 has a large contribution of $\sigma_{\boldsymbol{\phi}}(z_1) = 2.89$, which seems to make the character tall (see Fig. 4.4a). We decrease the parameter that modulates the contribution of PC 1 to $\sigma_{\boldsymbol{\phi}}(z_1) = -1.64$ to make the figure shorter. In the same way, we also increment the magnitude of PC 2 from $\sigma_{\boldsymbol{\phi}}(z_2) = 1.59$ to $\sigma_{\boldsymbol{\phi}}(z_2) = 2.99$, which appears to be correlated with the width of the character (refer to Fig. 4.4b). We display the modified meshes in Fig. 5.3 at (ii) and (iii).

Exchanging textures. Since all meshes share a common UV space, all textures can be exchanged between characters. Again, we select the pregnant woman in Fig. 5.1a as an example and apply the textures of the elf princess from Fig. 5.1b and John Cena from Fig. 5.1c. The results can be found in Fig. 5.3 at positions (iv) and (v).

5.5 Interchangeability of the Geometric Prior

One core limitation of our method, as compared to existing baselines, lies in our current focus on generating class-specific assets, which we demonstrate in Sec. 5.3 with human body shapes. Therefore, it is crucial that our generative framework is compatible with other geometric priors to extend its applicability to different classes. We verify the interchangeability of the geometric prior by using another 3DMM, namely the ICT-FaceKit [39]. This model is a highly detailed facial framework featuring physically-based attributes, quad-dominant topology, and tiled UV unwrapping.
For our experiment, we use the ICT Face Model Light, which includes 100 blend shapes for identity and 53 for expression. However, we chose to freeze the expression parameters, as our primary focus lies in generating the 3D face's shape and appearance. Importantly, we keep all the hyperparameters exactly the same as the ones reported in Sec. 5.3, ensuring that the compatibility is provided out of the box. Since we do not have an AABB available as with our own 3DMM for humans, we empirically set the limits to $a_i = -6.0$ and $b_i = 6.0$ for $i = 1, 2, \ldots, 100$. To better align the generated outputs with the facial focus of the model, we omit



(a) “a chubby Buddhist monk”

(b) “An ogre with a big nose”

(c) “A beautiful person”

(d) “Angela Merkel”

(e) “Dwayne Johnson”

Figure 5.4: Front, left, back, and right orthographic views of meshes synthesized with our method. Each view is rendered in MeshLab [11] with and without albedo texture to capture the geometric detail of the shape surface. The meshes are generated by prepending the string “portrait of ” and appending the string “, shaved head” to the labels in the captions.

the phrase “wearing a tank top” from the template prompt. The face model does not include the torso or extremities, which are prone to occlusion by non-tight or thick garments. Nevertheless, we still find it useful to prepend the phrase “portrait of ” and append “, shaved head” to guide the diffusion model to generate facial close-ups (as opposed to, e.g., full body images) and avoid the generation of hair, for which there is no geometry available in the face model. Example results are presented in Fig. 5.4, with additional samples in the extended version in Fig. 8.4 in Appendix 8.3.

We observe that our method seamlessly integrates with the ICT-FaceKit, achieving a high degree of perceptual quality and prompt alignment without requiring any modification to the framework. We find both the shape and the appearance of the output meshes to be faithful perceptual representations of the provided in-

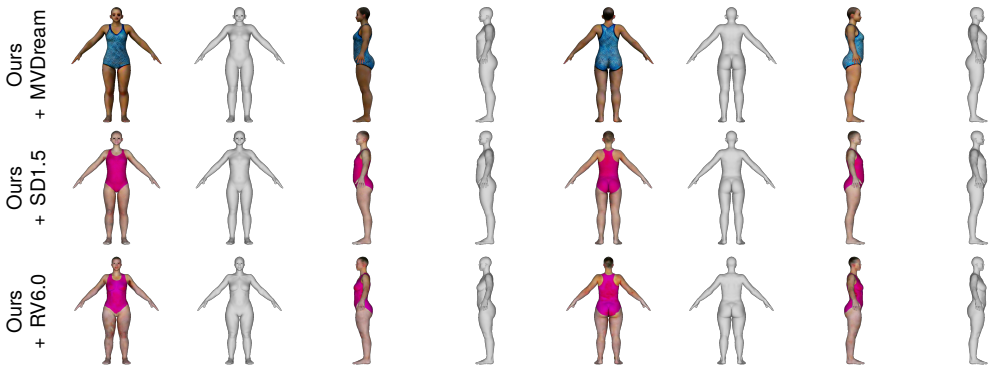
put prompt. These results align closely with our observations for full human body meshes, as discussed in Sec. 5.3. All of the outputs that we observe are 3D coherent and consistent from different perspectives, demonstrating that our method successfully surpasses the uncanny valley. The high quality of the provided topology and UVs provided by the ICT FaceKit make these meshes immediately suitable for production applications. However, we still find some undesired artifacts such as the shading being baked into the albedo map, blurry spots in the textures, and partially unrealistic body proportions, especially around the neck, as in Fig. 5.4c. While we have successfully demonstrated the interchangeability of the geometric prior using another 3DMM, there is no theoretical limitation preventing the application of other point distribution models. This flexibility highlights the broader potential of our approach across various domains.

5.6 Interchangeability of the Diffusion Prior

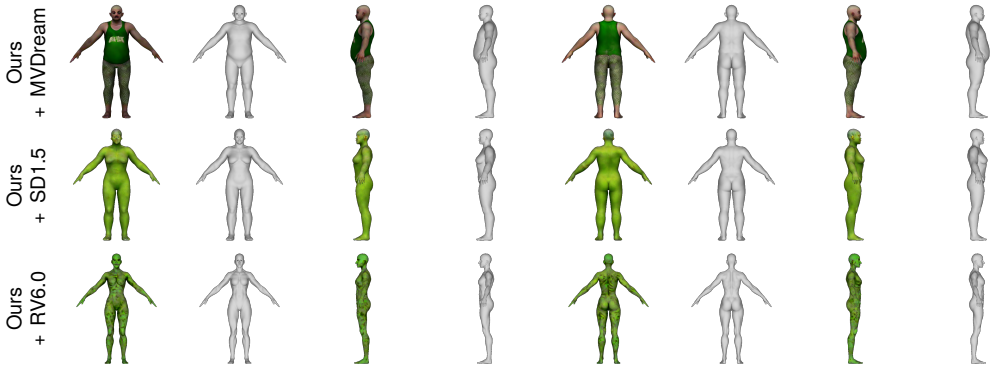
We point out that our method is also compatible with different diffusion priors. We find experimentally that MVDream [11] generally yields better results than single-view image diffusion models. However, this finding is not necessarily always the case. For instance, in Fig. 5.5, we ablate using StableDiffusion v1.5 [58] and RealisticVision v6.0 [64], which is a low rank adaptation (LoRA) [27] of StableDiffusion v1.5 finetuned for enhanced realism, instead of MVDream as the diffusion prior. Particularly, the examples shown in Fig. 5.5 provide meaningful and varied results that have arguably even better prompt alignment than the baseline in Fig. 5.5b. Potential use-cases of using a different diffusion prior go beyond the generation of variations (MVDream has a tendency to converge to similar modes across different runs with a fixed prompt, even when the seed is changed). We demonstrate in Fig. 5.5 how the style of the output is also affected by the choice of the diffusion model. Therefore, different LoRAs and models finetuned by the open-source community can be plugged into our 3D generative framework to achieve stylized results with a higher degree of control than prompt engineering. Another possible application involves alleviating undesired bias in the diffusion prior by using different checkpoints that address the issue.

5.7 Ablation Study

We ablate two choices that we make for the geometric prior. First, we show that AABB clipping is necessary for the SDS optimization to converge to shapes that adhere to the structural patterns of the 3DMM. Second, we also explore a different basis for the 3DMM that is not orthogonal and uncorrelated by doing interpolation with the raw blend shapes in the dataset instead of using the PCs.



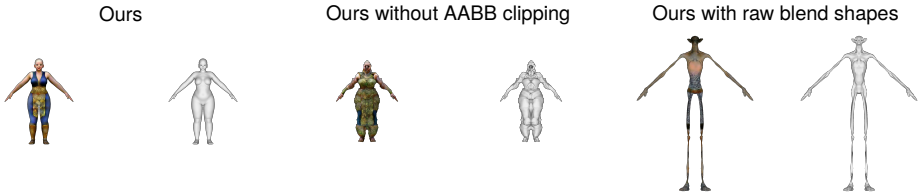
(a) Prompt: “slender bodied female aquatic gymnast, lean and athletic”.



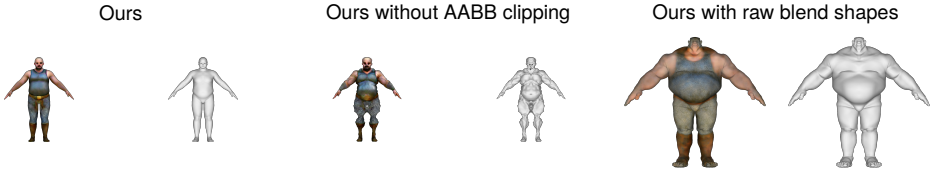
(b) Prompt: “very tall troll, ugly looking, green skin, lizard, humanoid”.

Figure 5.5: The effect of using different diffusion priors. From top row to bottom row, we utilize MVDream [67], StableDiffusion v1.5 (SD1.5) [58], and RealisticVision v6.0 (RV6.0) [64]. In each row, we display front, left, back, and right orthographic views of meshes synthesized with our method. Each view is rendered in MeshLab [11] with and without albedo texture to capture the geometric detail of the shape surface. The meshes are generated by appending the string “, shaved head, wearing a tank top” to the labels in the captions.

Removing AABB clipping. We find empirically that AABB clipping greatly improves the quality of the sampled meshes. As evidenced in Fig. 5.6, without AABB clipping, the sampled meshes still look similar to the ones in our baseline. Upon closer inspection, we see however that the surfaces are highly unsmooth and anatomically incorrect. We attribute this finding to the fact that the generative framework aims to stretch the available mesh surface to model accessories and items of clothing that are not captured by the 3DMM. In turn, we consider AABB clipping to better preserve the patterns of the data used to create the 3DMM. As we discuss in Appendix 8.1, there is no mathematical guarantee that AABB clip-



(a) "A Medieval Mongolian queen. She is obese and lives a sedentary life, but projects power and glory through her appearance"



(b) "A Medieval European King. He is obese and lives a sedentary life, but projects power and glory through his appearance"

Figure 5.6: We ablate the need of using AABB clipping and using a shape space that is not orthogonal and uncorrelated. On the left, we show results of our method as described in previous sections. At the center, we remove AABB clipping. On the right, we use raw blend shapes instead of the PC basis for the 3DMM. For each ablation, we present the textured rendering with albedo map and the untextured rendering. It is apparent that the performance of our approach deteriorates significantly with each ablation. All results are generated by appending the string ', shaved head, wearing a tank top" to the labels in the captions.

ping guides the SDS process to the manifold of the data: performing search in the shape space constrained by the AABB is not equivalent to seeking in the manifold of the dataset. The volume of the space defined by the AABB is several orders of magnitude larger than that of the convex hull of the data. We suspect that a considerable portion of the synthesized meshes extrapolates the dataset, although it is prohibitively expensive to explicitly compute the convex hull of our high dimensional dataset.

Using raw blend shapes instead of PCs. In Sec. 4.2.1, we make the case that performing PCA over the dataset, even without considerable dimensionality reduction, improves the optimization landscape because it spans an orthonormal basis of the dataset. We verify that the optimization landscape is considerably worse with a different basis by taking the raw dataset as a 3DMM. Instead of using the PCs, we span the shape space with the difference vectors of the raw blend shapes in the dataset with respect to the average mesh (i.e., the data matrix $\mathbf{\Delta S}$ in Sec. 4.2.1). In this case, the AABB trivially becomes $a_i = 0$ and $b_i = 1$ for $i = 1, 2, \ldots |D|$. As displayed in Fig. 5.6, the raw blend shape space is not suited for optimization via

gradient descent. Outputs become very exaggerated and shapes are driven to the extreme. We further observe that even minimal modifications in the prompt can lead to very different results, turning the model highly unpredictable.

6 Discussion

In this chapter, we discuss the implications of the presented results. We also evaluate the limitations of our method and explore future lines of research. Lastly, we examine the ethical considerations of our method.

6.1 Implications

Current research in text-to-3D generative models focuses on improving the quality of the images that are rendered from some 3D representation and on solving qualitative problems, such as the Janus problem and content drift. Since meshes do not offer the degree of flexibility that neural field representations do, which can differentiably render practically any 3D scene with a high degree of fidelity, most concurrent approaches choose neural fields similar to NeRFs [50] as their 3D representation. A mesh surface can then be extracted from the neural field with an isosurface algorithm. Even though mesh extraction is a destructive operation (due to the discretization of a continuous representation) and it recovers meshes with triangular and irregular topology, most of the metrics used to assess the quality of a 3D object operate in image space. It is hard to quantify the quality of the topology, skinning weights and UVs, or the perceptual quality of a 3D mesh as a whole. However, these features that cannot be measured are critical for real-life users, because game and rendering engines that are commercially available and widely adopted cannot render neural fields (with the exception of Gasusian Splatting) and rely on 3D meshes. Therefore, there is a gap between academic research and industry needs.
In this work, we bridge the aforementioned disconnect between academia and industry and propose a generative framework that strikes a trade-off between flexibility and output quality. By using a 3DMM as a geometric prior, we make it possible to sample meshes with artist-designed topology, skinning weights, and UVs. Thus, our model can sample operational meshes at the cost of being restricted to the shape space of the 3DMM. Furthermore, the 3DMM also acts as a prior over the shape space, resulting in meshes that exhibit a high level of detail and realism in their shapes. For color, we use a neural field that is versatile and is not limited in texture space. In our quantitative analysis, we find that our method outperforms all previous baselines on multi-view prompt alignment. In addition, we observe qualitatively that the output meshes from our approach have smoother surfaces, and

are anatomically accurate and consistent across different views. We also motivate the choice of a low CFG scale that is close to the optimal value used for image synthesis, which solves the problem of over-saturated colors in textures.
Beyond the results that we show in Chapter 5, we also highlight one more unique advantage of our generative framework, which is consistency. Not only is consistency important to ensure stylistic coherence within a compositional scene, but also for technical reasons. All output meshes share the same topology for a fixed 3DMM, which means that all accessories that an artists builds for one of the meshes can be shared across all meshes. For example, a garment that tightly fits one body shape can be fit to another body shape by interpolating the deformation field between the point clouds of the two body meshes on the point cloud of the garment. In the same way, the UV space and skinning weights are shared between meshes, allowing to easily apply different textures and animations from a shared library.

6.2 Limitations and Future Work

Our method is capable of synthesizing production-ready meshes, but there are still some limitations that need to be addressed in future work to enhance the usefulness of the model. We consider issues that limit the range of applications of our generative pipeline for widespread adoption and propose approaches to resolve them.

The model is constrained by the expressivity of the 3DMM. Our generative model is restricted to the shape space of the 3DMM. It is not possible, for instance, to prompt the model to create a bird when the 3DMM represents humanoids. Nevertheless, we show in Sec. 5.5 that the geometric prior can be interchanged with other 3DMMs without requiring any modification to the framework or the hyperparameters. Thus, a simple preprocessing step could be added that classifies the text prompt via natural language processing to use one out of a set of available 3DMMs. Building new 3DMMs has recently become more accessible with mesh registration algorithms [4, 45] that allow to register a template topology to a dataset of meshes that have inconsistent topology, such as scanned 3D objects. It would also be highly valuable to extend our 3DMM to also capture garments and polygonal hair, potentially as separate mesh components.

Output meshes are not fully rigged. Meshes have to be rigged in order to be animatable. The rig is composed mainly of two components: a skeleton and a set of skinning weights. The skeleton defines a bone hierarchy that determines the pose of the mesh. All vertices have a skinning weight for each bone, which encodes the influence of each bone's translation and rotation (relative to the resting

to pose). The position of the vertex can then be computed via linear blend skinning or dual quaternion skinning [31]. Since the skinning weights are vertex attributes and the connectivity of the output mesh is fully characterized by the 3DMM, the skinning weights can be specified by an artist at design time. However, the position of the bones has to be predicted for the output mesh to be fully rigged, which is a regression problem that can be solved given enough labeled data.

Textures are not PBR. Our method is able to predict albedo maps with realistic colors, although it does not achieve photorealism. In addition, it is still possible that the texture represents features that do not correspond in space with the underlying shape of the mesh, as seen in Fig. 5.4 around the eyes of some examples (e.g., Angela Merkel's eyes appear to be painted below the geometry reserved for the eyeballs). Furthermore, the albedo map still portrays some shading effects. In follow-up work, we suggest to include available PBR textures in the 3DMM, as in [14]. This way, the geometric prior would define a prior not only over the shape space and the geometrical structure of the mesh graph, but also over the texture space. In this scenario, the differentiable renderer can be simplified to directly sample the color from the UV space instead of using the feature network with multi-resolution has encoding. Further adaptations need be made to support PBR effects differentiably via [37] to efficiently search within the blend space of PBR textures.

Generation time is slow. As mentioned in Sec. 5.1, it currently takes almost 20 minutes for SDS to converge over 7000 iterations on an Nvidia Tesla A100 GPU. While the generation time is still considerably faster than all our baselines, which can take hours on the same hardware, we still see room for improvement. The reason why we need the slow SDS optimization is that diffusion models are inconsistent from multiple views, which prevents us from directly fitting the mesh via reconstruction loss with a set of diffused images that cover the entire visible surface of the 3D object. Architectures like SyncDreamer [44] and MVDream [67] boost the multi-view consistency of the diffusion model, but the issue remains and it is still not possible to directly fit the 3D mesh in a single iteration. In principle, the mechanism that is needed for multi-view consistency in diffusion models is the same that enables spatio-temporal consistency in video diffusion models. The problem of diffusing multiple views of an object is equivalent to diffusing a video where the camera rotates around the object. For example, Chen, Marti Monso, et. al [10] propose a training paradigm for diffusion models that shows great promise for autoregressive video synthesis. We could use that approach to train a diffusion prior with strong multi-view consistency that enables single-step reconstruction for fast mesh synthesis, similar to Cat3D [16].

Language. The diffusion priors that we use are trained on English captions and may not work in other languages. We demonstrate in Sec. 5.6 that the diffusion prior can be interchanged, which allows using diffusion models trained on other languages.

6.3 Ethics

Dataset copyright. The dataset that we use to create our 3DMM is a proprietary dataset that is fully licensed. All remaining models that we evaluate (3DMMs and diffusion priors) have also been trained on licensed datasets.

Safety. Our 3D generative model inherits the capability of the diffusion prior to generate NSFW content. We advise to use the model responsibly. Optionally, available NSFW filters can be enabled at the cost of potentially hurting model performance.

Bias and fairness. As we discuss in Sec. 5.3, our method also takes over the bias of the diffusion priors that we use. We observe that the model captures social bias present in cultures that use the English language, because the diffusion models are trained on datasets with English captions.

Environment. We do not train any diffusion model ourselves and leverage pretrained components only. Inference takes about 20 minutes on an A100 SXM4 80GB and we use Google Cloud Platform as our cloud provider on the us-central1 compute region. Using the Machine Learning Emissions Calculator from [36], we estimate $0.07\,\mathrm{kg}$ CO_2 eq. per sampled mesh. These emissions are entirely offset by the cloud provider.

Impact on employment. Generative models are often perceived as a threat to the labor market. We do not aim to substitute technical artists or creators, but rather to empower their workflows by providing cost-efficient tools that assist them with technical tasks and enable them to unleash their creativity. Media creators and artists can spend more time designing art concepts, interactions, and processes without being hindered by the deep 3D modeling skills required to generate 3D assets. We believe that more people will be able to participate in the industry of 3D content creation by lowering the barrier of entry, which will drive progress and new developments across various sectors in the entertainment sector and beyond.

7 Conclusion

In this work, we address the core limitation of text-to-3D mesh generative models to be useful in production settings. We make the observation that 3D neural field representations are not suited to represent the structural properties of mesh surfaces. Instead, we propose a generative model that directly samples high-quality 3D meshes by combining a 3DMM with a multi-view diffusion prior. The SDS guidance signal is backpropagated through a differentiable mesh renderer into the shape space of the 3DMM and into an MLP with multi-resolution hash encoding for point color.

We implement our method around our own 3DMM that we derive from a dataset of humanoid blend shapes. Then, we benchmark our approach against a set of baselines and demonstrate higher correspondence between the input prompt (constrained to humanoids) and the output mesh, measured on the MV-CLIP score that interpolates the CLIP image embeddings across different views to account for multi-view coherence. The meshes sampled with our generative framework have artist-designed quadrangular topology, UVs, and skinning weights. Moreover, the shapes display a high degree of realism and anisotropic surface detail. We justify that our method can run SDS with optimal CFG scale for image synthesis because the geometric prior restricts the shape space in a way that facilitates mode seeking. As a consequence of the lower CFG coefficient, the colors in our albedo textures are not oversaturated, as is usual in SDS models.

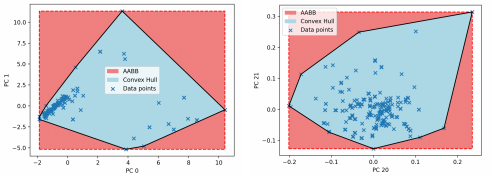
One trade-off of our generative model is that it is limited to prompts and assets that can be represented in the shape space of the 3DMM at use. Thus, we show that our framework is compatible with other 3DMMs (e.g., for human faces) and diffusion priors, which enhances the versatility of the model. Furthermore, output meshes are interpretable in the sense that the output geometry can be fully explained by the parameter vector that controls the contribution of each PC. Therefore, the mesh can be easily edited by manually adjusting the coefficients without affecting the texture. Similarly, textures can be exchanged between sampled outputs, as the meshes share a common UV space. At last, we also ablate some choices that we make for the geometric prior, such as AABB clipping and the selection of an orthogonal and uncorrelated basis for the shape space.

Finally, we highlight possible improvements for follow-up work that would amplify the usability of output meshes, such as including PBR materials in the 3DMM, adding garments and hair to the 3DMM, regressing the skeletal rig, or using video diffusion architectures that strengthen the multi-view consistency of the diffusion

prior for faster predictions. All in all, we bridge a critical disconnect between academic research and user needs, and aim to contribute to the adoption of automated mesh generation workflows in real-life applications to alleviate the burden on technical artists and lower the barrier of entry to 3D content creation, potentially enabling novel use cases.

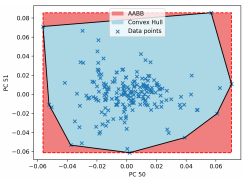
8 Appendix

8.1 Analysis of the Shape Space Constrained by the Axis-Aligned Bounding Box

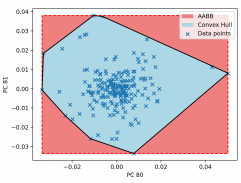


(a) PCs 0 and 1.

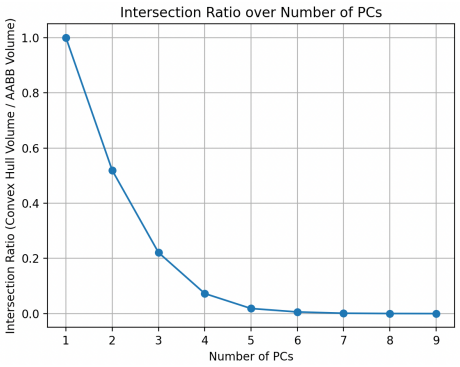
(b) PCs 20 and 21.



(c) PCs 50 and 51.



(d) PCs 80 and 81.



(e) Cumulative intersection ratio between the volume of the convex hull and the volume of the AABB plotted over the number of PCs.

Figure 8.1: We analyze the space occupied by the convex hull and that of the AABB of the data when projected to the shape space spanned by the PCs. Figs. 8.1a–8.1d show both manifolds and the contained data points plotted over two dimensions (note the zero-base encoding of the labels). Figure 8.1e displays how the intersection ratio between the two volumes decreases exponentially as the number of PCs increases.

We make the empirical observation that the intersecting volume between the convex hull of the dataset projected to the PC space and its AABB converges to 0 extremely rapidly. Despite the fact that the intersection area seems to be large for any two randomly selected dimensions (Figs. 8.1a–8.1d), the intersection ratio of the two volumes is practically 0 after adding as little as seven dimensions (Fig. 8.1e). Computing the actual intersection volume of the full 100-dimensional space becomes intractable.
An intuitive geometrical explanation could be that the convex polytope defined by the convex hull is contained by a sphere of finite radius. The volume of an n-ball of fixed radius tends to a limiting value of 0 as n goes to infinity [76]. However, we find experimentally that the volume of the convex hull diverges, suggesting that the rea-

soning does not apply. Instead, we note that for any two PCs $i, j \in \{1, 2, \ldots, 100\}$ with $i \neq j$, the intersection ratio $\alpha_{i,j}$ between the convex hull area and the AABB area is bound by $0.379959 \leq \alpha_{i,j} \leq 0.887392$. Thus, the intersection ratio over the number of dimensions n is lower bound by the function 0.379959^n and upper bound by 0.887392^n, showing that the decay is truly exponential. For $n = 100$, we find that there is a difference of at least 6 orders of magnitude (taking the upper bound) and up to 43 orders of magnitude (taking the lower bound) between the two volumes. Although both values are subject to numerical errors and potentially inaccurate, we show that most of the shape space of the geometric prior is extrapolating with respect to the original data and only an almost negligible portion is actually interpolating.

8.2 Derivation of the Score Distillation Sampling Decomposition

Here, we derive the decomposition of the SDS gradient $\nabla_z \mathcal{L}_{\text{SDS}}$ in Eq. (4.13). We start by rewriting the classifier-free guidance from (3.10):

$$\begin{aligned}
\hat{\boldsymbol{\varepsilon}}_{\boldsymbol{\phi}}(\boldsymbol{x}_t; y, t) &= (1 + w_{\text{CFG}})\boldsymbol{\varepsilon}_{\boldsymbol{\phi}}(\boldsymbol{x}_t; y, t) - w_{\text{CFG}}\boldsymbol{\varepsilon}_{\boldsymbol{\phi}}(\boldsymbol{x}_t; t) && \text{(8.1a)}\\
&= w_{\text{CFG}}\left(\boldsymbol{\varepsilon}_{\boldsymbol{\phi}}(\boldsymbol{x}_t; y, t) - \boldsymbol{\varepsilon}_{\boldsymbol{\phi}}(\boldsymbol{x}_t; t)\right) + \boldsymbol{\varepsilon}_{\boldsymbol{\phi}}(\boldsymbol{x}_t; y, t) && \text{(8.1b)}\\
&= (w_{\text{CFG}} + 1)\left(\boldsymbol{\varepsilon}_{\boldsymbol{\phi}}(\boldsymbol{x}_t; y, t) - \boldsymbol{\varepsilon}_{\boldsymbol{\phi}}(\boldsymbol{x}_t; t)\right) + \boldsymbol{\varepsilon}_{\boldsymbol{\phi}}(\boldsymbol{x}_t; y, t)\\
&\quad - \left(\boldsymbol{\varepsilon}_{\boldsymbol{\phi}}(\boldsymbol{x}_t; y, t) - \boldsymbol{\varepsilon}_{\boldsymbol{\phi}}(\boldsymbol{x}_t; t)\right) && \text{(8.1c)}\\
&= (w_{\text{CFG}} + 1)\left(\boldsymbol{\varepsilon}_{\boldsymbol{\phi}}(\boldsymbol{x}_t; y, t) - \boldsymbol{\varepsilon}_{\boldsymbol{\phi}}(\boldsymbol{x}_t; t)\right) + \boldsymbol{\varepsilon}_{\boldsymbol{\phi}}(\boldsymbol{x}_t; t). && \text{(8.1d)}
\end{aligned}$$

The initial expression (8.1a) is the equation for classifier-free guidance (3.10) as originally formulated by [23]. Then, step (8.1b) follows by the distributive property and in (8.1c) we add the zero-term

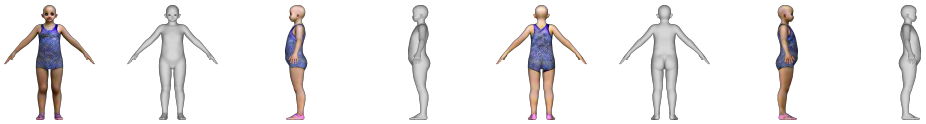
$$+\left(\boldsymbol{\varepsilon}_{\boldsymbol{\phi}}(\boldsymbol{x}_t; y, t) - \boldsymbol{\varepsilon}_{\boldsymbol{\phi}}(\boldsymbol{x}_t; t)\right) - \left(\boldsymbol{\varepsilon}_{\boldsymbol{\phi}}(\boldsymbol{x}_t; y, t) - \boldsymbol{\varepsilon}_{\boldsymbol{\phi}}(\boldsymbol{x}_t; t)\right) = 0.$$

Next, we simplify the expression in (8.1d) by cancellation. Finally, we insert the newly obtained expression for the classifier-free guidance $\hat{\boldsymbol{\varepsilon}}_{\boldsymbol{\phi}}(\boldsymbol{x}_t; y, t)$ into the equation for the SDS gradient (3.11):

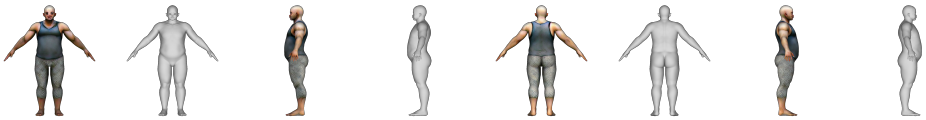
$$\begin{aligned}
\nabla_z \mathcal{L}_{\text{SDS}} &= \mathbb{E}_{t,\boldsymbol{\varepsilon}}\left[w(t)\left(\hat{\boldsymbol{\varepsilon}}_{\boldsymbol{\phi}}(\boldsymbol{x}_t; y, t) - \boldsymbol{\varepsilon}\right)\frac{\partial \boldsymbol{x}_t}{\partial z}\right] && \text{(8.2a)}\\
&= \mathbb{E}_{t,\boldsymbol{\varepsilon}}\Bigg[w(t)\left((w_{\text{CFG}} + 1)\left(\boldsymbol{\varepsilon}_{\boldsymbol{\phi}}(\boldsymbol{x}_t; y, t) - \boldsymbol{\varepsilon}_{\boldsymbol{\phi}}(\boldsymbol{x}_t; t)\right)\right)\frac{\partial \boldsymbol{x}_t}{\partial z}\\
&\qquad + w(t)\left(\boldsymbol{\varepsilon}_{\boldsymbol{\phi}}(\boldsymbol{x}_t; t) - \boldsymbol{\varepsilon}\right)\frac{\partial \boldsymbol{x}_t}{\partial z}\Bigg] && \text{(8.2b)}\\
&= \nabla_z \mathcal{L}_{\text{proj}} + (w_{\text{CFG}} + 1)\,\nabla_z \mathcal{L}_{\text{cond}} \quad . && \text{(8.2c)}
\end{aligned}$$

Again, the initial expression (8.2a) is the equation for the SDS gradient (3.11) as originally formulated by [56]. In step (8.2b) we insert Eq. (8.1d) and apply the distributive property. We obtain Eq. (8.2c), which is equivalent to Eq. (4.13), by the linearity of expectation (and the fact that w_{CFG} is constant), the substitution of $\mathcal{L}_{\text{proj}}$ from (4.13a) and $\mathcal{L}_{\text{cond}}$ from (4.13b), and the commutative property.

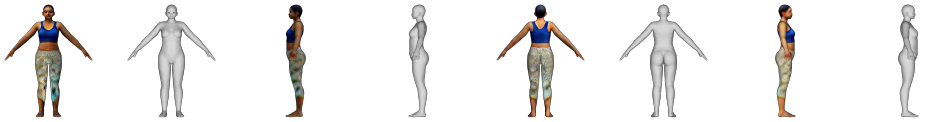
8.3 Extended Results



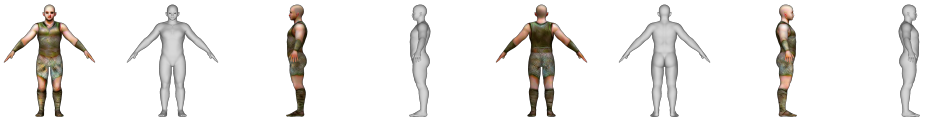
(a) “6 year old girl, lean, gymnast, happy, rosy cheeks”



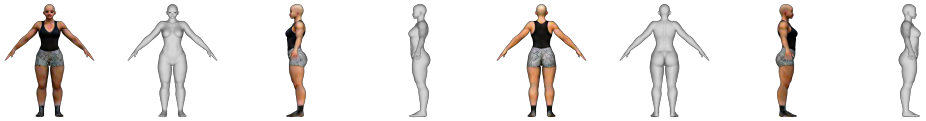
(b) “dad body”



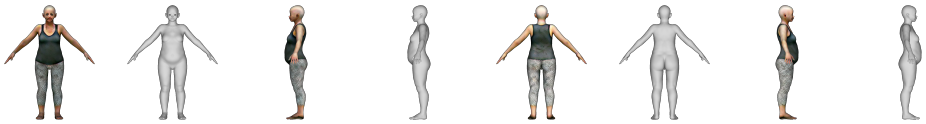
(c) “A woman from southern India, she is very traditional, hinduist, 43 years old, serene expression”



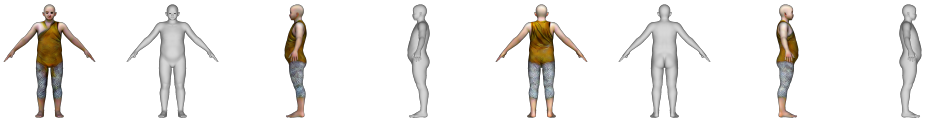
(d) “A very young Roman soldier wearing his armor. He is short, weak, and inexperienced, you can tell he is not ready for the battlefield.”



(e) “a female bodybuilder, muscular, buffed, strong and femenine”

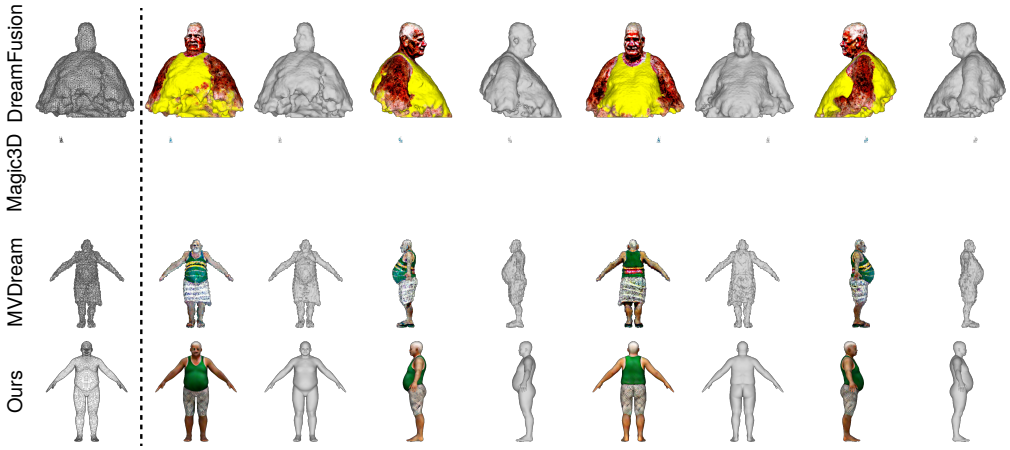


(f) “Very old woman”



(g) “an 18 year old Buddhist monk, childish looking”

Figure 8.2: Extended results for Fig. 5.1. Front, left, back, and right orthographic views of meshes synthesized with our method. Each view is rendered in MeshLab [11] with and without albedo texture to capture the geometric detail of the shape surface. The meshes are generated by appending the string “, shaved head, wearing a tank top” to the labels in the captions.



(a) “A portly Aboriginal Australian, 80 years old, very wise”

(b) “a male bodybuilder, muscular, buffed, strong and masculine”

Figure 8.3: Extended results for Fig. 5.2. We benchmark our method (fourth row) against DreamFusion [56] (first row), Magic3D [41] (second row), and MVDream [67] (third row). As seen in the first column, ours is the only approach capable of synthesizing meshes with clean topology. The remaining eight columns show front, left, back, and right orthographic views rendered in MeshLab [11] with and without albedo texture. Our method is the only that produces smooth surfaces and realistic body anatomy. All meshes are generated by appending the string “, shaved head, wearing a tank top” to the labels in the captions.

(a) “Serena Williams”

(b) “Jackie Chan”

(c) “A man with a long face, high forehead”

(d) “A highly asymmetrical face”

(e) “a very old lady, saggy skin, double chin”

(f) “Homer Simpson”

(g) “A baby”

Figure 8.4: Extended results for Fig. 5.4. Front, left, back, and right orthographic views of meshes synthesized with our method. Each view is rendered in MeshLab [11] with and without albedo texture to capture the geometric detail of the shape surface. The meshes are generated by appending the string “, shaved head, wearing a tank top” to the labels in the captions.

Bibliography

[1] T. Alldieck, N. Kolotouros, and C. Sminchisescu. *Score Distillation Sampling with Learned Manifold Corrective*. 2024. arXiv: 2401.05293 [cs.CV].

[2] M. Armandpour, H. Zheng, A. Sadeghian, A. Sadeghian, and M. Zhou. "Re-imagine the Negative Prompt Algorithm: Transform 2D Diffusion into 3D, alleviate Janus problem and Beyond". In: *arXiv preprint arXiv:2304.04968* (2023).

[3] D. Bahdanau, K. Cho, and Y. Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2016. arXiv: 1409.0473 [cs.CL].

[4] M. Bahri, E. O. Sullivan, S. Gong, F. Liu, X. Liu, M. M. Bronstein, and S. Zafeiriou. *Shape My Face: Registering 3D Face Scans by Surface-to-Surface Translation*. 2021. arXiv: 2012.09235 [cs.CV]. URL: https://arxiv.org/abs/2012.09235.

[5] Y. Balaji, S. Nah, X. Huang, A. Vahdat, J. Song, Q. Zhang, K. Kreis, M. Aittala, T. Aila, S. Laine, B. Catanzaro, T. Karras, and M.-Y. Liu. "eDiff-I: Text-to-Image Diffusion Models with Ensemble of Expert Denoisers". In: *arXiv preprint arXiv:2211.01324* (2022).

[6] A. Bansal, E. Borgnia, H.-M. Chu, J. S. Li, H. Kazemi, F. Huang, M. Goldblum, J. Geiping, and T. Goldstein. *Cold Diffusion: Inverting Arbitrary Image Transforms Without Noise*. 2022. arXiv: 2208.09392 [cs.CV].

[7] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan. "Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields". In: *ICCV* (2021).

[8] V. Blanz and T. Vetter. "A Morphable Model for the Synthesis of 3D Faces". In: *SIGGRAPH'99 Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (1999). DOI: 10.1145/311535.311556.

[9] A. X. Chang, T. Funkhouser, L. 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*. 2015. arXiv: 1512.03012 [cs.GR]. URL: https://arxiv.org/abs/1512.03012.

[10] B. Chen, D. M. Monso, Y. Du, M. Simchowitz, R. Tedrake, and V. Sitzmann. *Diffusion Forcing: Next-token Prediction Meets Full-Sequence Diffusion*. 2024. arXiv: 2407.01392 [cs.LG]. URL: https://arxiv.org/abs/2407.01392.

[11] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia. "MeshLab: an Open-Source Mesh Processing Tool". In: *Eurographics Italian Chapter Conference*. Ed. by V. Scarano, R. D. Chiara, and U. Erra. The Eurographics Association, 2008. ISBN: 978-3-905673-68-5. DOI: 10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136.

[12] B. O. Community. *Blender - a 3D modelling and rendering package*. Blender Foundation. Stichting Blender Foundation, Amsterdam, 2018. URL: http://www.blender.org.

[13] M. Deitke, R. Liu, M. Wallingford, H. Ngo, O. Michel, A. Kusupati, A. Fan, C. Laforte, V. Voleti, S. Y. Gadre, E. VanderBilt, A. Kembhavi, C. Vondrick, G. Gkioxari, K. Ehsani, L. Schmidt, and A. Farhadi. *Objaverse-XL: A Universe of 10M+ 3D Objects*. 2023. arXiv: 2307.05663 [cs.CV]. URL: https://arxiv.org/abs/2307.05663.

[14] B. Egger, W. A. P. Smith, A. Tewari, S. Wuhrer, M. Zollhoefer, T. Beeler, F. Bernard, T. Bolkart, A. Kortylewski, S. Romdhani, C. Theobalt, V. Blanz, and T. Vetter. *3D Morphable Face Models – Past, Present and Future*. 2020. arXiv: 1909.01815 [cs.CV]. URL: https://arxiv.org/abs/1909.01815.

[15] J. Gao, T. Shen, Z. Wang, W. Chen, K. Yin, D. Li, O. Litany, Z. Gojcic, and S. Fidler. "GET3D: A Generative Model of High Quality 3D Textured Shapes Learned from Images". In: *Advances In Neural Information Processing Systems*. 2022.

[16] R. Gao*, A. Holynski*, P. Henzler, A. Brussee, R. Martin-Brualla, P. P. Srinivasan, J. T. Barron, and B. Poole*. "CAT3D: Create Anything in 3D with Multi-View Diffusion Models". In: *arXiv* (2024).

[17] *GitHub - jpcy/xatlas: Mesh parameterization / UV unwrapping library — github.com*. https://github.com/jpcy/xatlas. [Accessed 03-09-2024].

[18] Y.-C. Guo, Y.-T. Liu, R. Shao, C. Laforte, V. Voleti, G. Luo, C.-H. Chen, Z.-X. Zou, C. Wang, Y.-P. Cao, and S.-H. Zhang. *threestudio: A unified framework for 3D content generation*. https://github.com/threestudio-project/threestudio. 2023.

[19] N. Guttenberg. *Diffusion with Offset Noise*. 2023. URL: https://www.crosslabs.org/blog/diffusion-with-offset-noise.

[20] A. Hertz, K. Aberman, and D. Cohen-Or. *Delta Denoising Score*. 2023. arXiv: 2304.07090 [cs.CV]. URL: https://arxiv.org/abs/2304.07090.

[21] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. *GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium*. 2018. arXiv: 1706.08500 [cs.LG].

[22] J. Ho, A. Jain, and P. Abbeel. "Denoising Diffusion Probabilistic Models". In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 6840–6851. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf.

[23] J. Ho and T. Salimans. "Classifier-Free Diffusion Guidance". In: *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*. 2021.

[24] J. Ho, T. Salimans, A. Gritsenko, W. Chan, M. Norouzi, and D. J. Fleet. *Video Diffusion Models*. 2022. arXiv: 2204.03458 [cs.CV]. URL: https://arxiv.org/abs/2204.03458.

[25] F. Hong, M. Zhang, L. Pan, Z. Cai, L. Yang, and Z. Liu. "AvatarCLIP: Zero-Shot Text-Driven Generation and Animation of 3D Avatars". In: *ACM Transactions on Graphics (TOG)* 41(4) (2022), pp. 1–19.

[26] Y. Hong, K. Zhang, J. Gu, S. Bi, Y. Zhou, D. Liu, F. Liu, K. Sunkavalli, T. Bui, and H. Tan. *LRM: Large Reconstruction Model for Single Image to 3D*. 2024. arXiv: 2311.04400 [cs.CV]. URL: https://arxiv.org/abs/2311.04400.

[27] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. *LoRA: Low-Rank Adaptation of Large Language Models*. 2021. arXiv: 2106.09685 [cs.CL]. URL: https://arxiv.org/abs/2106.09685.

[28] Y. Huang, J. Wang, Y. Shi, X. Qi, Z.-J. Zha, and L. Zhang. "DreamTime: An Improved Optimization Strategy for Text-to-3D Content Creation". In: (2023). arXiv: 2306.12422 [cs.CV].

[29] Y. Huang, J. Wang, A. Zeng, H. Cao, X. Qi, Y. Shi, Z.-J. Zha, and L. Zhang. "DreamWaltz: Make a Scene with Complex 3D Animatable Avatars". In: (2023). arXiv: 2305.12529 [cs.CV].

[30] A. Jain, B. Mildenhall, J. T. Barron, P. Abbeel, and B. Poole. *Zero-Shot Text-Guided Object Generation with Dream Fields*. 2022. arXiv: 2112.01455 [cs.CV]. URL: https://arxiv.org/abs/2112.01455.

[31] L. Kavan. “Part I: Direct Skinning Methods and Deformation Primitives”. In: 2014. URL: https://api.semanticscholar.org/CorpusID:7004409.

[32] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis. “3D Gaussian Splatting for Real-Time Radiance Field Rendering”. In: *ACM Transactions on Graphics* 42(4) (July 2023). URL: https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/.

[33] D. P. Kingma and M. Welling. *Auto-Encoding Variational Bayes*. 2013. arXiv: 1312.6114 [stat.ML]. URL: https://arxiv.org/abs/1312.6114.

[34] D. P. Kingma and J. Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG]. URL: https://arxiv.org/abs/1412.6980.

[35] D. P. Kingma and M. Welling. “An Introduction to Variational Autoencoders”. In: *Foundations and Trends® in Machine Learning* 12(4) (2019), pp. 307–392. DOI: 10.1561/2200000056. URL: https://doi.org/10.1561%2F2200000056.

[36] A. Lacoste, A. Luccioni, V. Schmidt, and T. Dandres. *Quantifying the Carbon Emissions of Machine Learning*. 2019. arXiv: 1910.09700 [cs.CY]. URL: https://arxiv.org/abs/1910.09700.

[37] S. Laine, J. Hellsten, T. Karras, Y. Seol, J. Lehtinen, and T. Aila. “Modular Primitives for High-Performance Differentiable Rendering”. In: *ACM Transactions on Graphics* 39(6) (2020).

[38] J. Lambert. *Photometria sive de mensura et gradibus luminis, colorum et umbrae*. sumptibus vidvae E. Klett, typis C.P. Detleffsen, 1760. URL: https://books.google.es/books?id=JdkTAAAAQAAJ.

[39] R. Li, K. Bladin, Y. Zhao, C. Chinara, O. Ingraham, P. Xiang, X. Ren, P. Prasad, B. Kishore, J. Xing, and H. Li. *Learning Formation of Physically-Based Face Attributes*. 2020. arXiv: 2004.03458 [cs.CV].

[40] X. Li, Q. Zhang, D. Kang, W. Cheng, Y. Gao, J. Zhang, Z. Liang, J. Liao, Y.-P. Cao, and Y. Shan. *Advances in 3D Generation: A Survey*. 2024. arXiv: 2401.17807 [cs.CV]. URL: https://arxiv.org/abs/2401.17807.

[41] C.-H. Lin, J. Gao, L. Tang, T. Takikawa, X. Zeng, X. Huang, K. Kreis, S. Fidler, M.-Y. Liu, and T.-Y. Lin. *Magic3D: High-Resolution Text-to-3D Content Creation*. 2023. arXiv: 2211.10440 [cs.CV]. URL: https://arxiv.org/abs/2211.10440.

[42] S. Lin, B. Liu, J. Li, and X. Yang. *Common Diffusion Noise Schedules and Sample Steps are Flawed*. 2023. arXiv: 2305.08891 [cs.CV].

[43] R. Liu, R. Wu, B. V. Hoorick, P. Tokmakov, S. Zakharov, and C. Vondrick. *Zero-1-to-3: Zero-shot One Image to 3D Object*. 2023. arXiv: 2303.11328 [cs.CV].

[44] Y. Liu, C. Lin, Z. Zeng, X. Long, L. Liu, T. Komura, and W. Wang. "SyncDreamer: Generating Multiview-consistent Images from a Single-view Image". In: *The Twelfth International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=MN3yH2ovHb.

[45] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. "SMPL: A Skinned Multi-Person Linear Model". In: *ACM Trans. Graphics (Proc. SIGGRAPH Asia)* 34(6) (Oct. 2015), 248:1–248:16.

[46] W. E. Lorensen and H. E. Cline. "Marching cubes: A high resolution 3D surface construction algorithm". In: *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '87. New York, NY, USA: Association for Computing Machinery, 1987, pp. 163–169. ISBN: 0897912276. DOI: 10.1145/37401.37422. URL: https://doi.org/10.1145/37401.37422.

[47] I. Loshchilov and F. Hutter. *Decoupled Weight Decay Regularization*. 2019. arXiv: 1711.05101 [cs.LG]. URL: https://arxiv.org/abs/1711.05101.

[48] A. Lukoianov, H. S. de Ocáriz Borde, K. Greenewald, V. C. Guizilini, T. Bagautdinov, V. Sitzmann, and J. Solomon. *Score Distillation via Reparametrized DDIM*. 2024. arXiv: 2405.15891 [cs.CV]. URL: https://arxiv.org/abs/2405.15891.

[49] M.-T. Luong, H. Pham, and C. D. Manning. *Effective Approaches to Attention-based Neural Machine Translation*. 2015. arXiv: 1508.04025 [cs.CL].

[50] 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*. 2020. arXiv: 2003.08934 [cs.CV]. URL: https://arxiv.org/abs/2003.08934.

[51] T. Müller. *tiny-cuda-nn*. Version 1.7. Apr. 2021. URL: https://github.com/NVlabs/tiny-cuda-nn.

[52] T. Müller, A. Evans, C. Schied, and A. Keller. "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding". In: *ACM Trans. Graph.* 41(4) (July 2022), 102:1–102:15. DOI: 10.1145/3528223.3530127. URL: https://doi.org/10.1145/3528223.3530127.

[53] A. Q. Nichol and P. Dhariwal. “Improved Denoising Diffusion Probabilistic Models”. In: *Proceedings of the 38th International Conference on Machine Learning*. Vol. 139. Proceedings of Machine Learning Research. PMLR, July 2021, pp. 8162–8171. URL: https://proceedings.mlr.press/v139/nichol21a.html.

[54] D. H. Park, S. Azadi, X. Liu, T. Darrell, and A. Rohrbach. “Benchmark for Compositional Text-to-Image Synthesis”. In: *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*. Ed. by J. Vanschoren and S. Yeung. Vol. 1. 2021.

[55] A. Pelykh, O. M. Sincan, and R. Bowden. *Giving a Hand to Diffusion Models: a Two-Stage Approach to Improving Conditional Human Image Generation*. 2024. arXiv: 2403.10731 [cs.CV]. URL: https://arxiv.org/abs/2403.10731.

[56] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall. “DreamFusion: Text-to-3D using 2D Diffusion”. In: *arXiv* (2022).

[57] 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*. 2021. arXiv: 2103.00020 [cs.CV].

[58] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2021. arXiv: 2112.10752 [cs.CV].

[59] O. Ronneberger, P. Fischer, and T. Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Ed. by N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi. Cham: Springer International Publishing, 2015, pp. 234–241.

[60] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. Denton, S. K. S. Ghasemipour, B. K. Ayan, S. S. Mahdavi, R. G. Lopes, T. Salimans, J. Ho, D. J. Fleet, and M. Norouzi. *Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding*. 2022. arXiv: 2205.11487 [cs.CV]. URL: https://arxiv.org/abs/2205.11487.

[61] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. *Improved Techniques for Training GANs*. 2016. arXiv: 1606.03498 [cs.LG]. URL: https://arxiv.org/abs/1606.03498.

[62] T. Salimans and J. Ho. “Progressive Distillation for Fast Sampling of Diffusion Models”. In: *International Conference on Learning Representations*. 2022.

[63] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, P. Schramowski, S. Kundurthy, K. Crowson, L. Schmidt, R. Kaczmarczyk, and J. Jitsev. *LAION-5B: An open large-scale dataset for training next generation image-text models*. 2022. arXiv: 2210.08402 [cs.CV]. URL: https://arxiv.org/abs/2210.08402.

[64] SG161222. *Realistic Vision V6.0 B1 noVAE*. Accessed: 2024-08-27. Aug. 2024. URL: https://huggingface.co/SG161222/Realistic_Vision_V6.0_B1_noVAE.

[65] T. Shen, J. Gao, K. Yin, M.-Y. Liu, and S. Fidler. *Deep Marching Tetrahedra: a Hybrid Representation for High-Resolution 3D Shape Synthesis*. 2021. arXiv: 2111.04276 [cs.CV]. URL: https://arxiv.org/abs/2111.04276.

[66] T. Shen, J. Munkberg, J. Hasselgren, K. Yin, Z. Wang, W. Chen, Z. Gojcic, S. Fidler, N. Sharp, and J. Gao. "Flexible Isosurface Extraction for Gradient-Based Mesh Optimization". In: *ACM Trans. Graph.* 42(4) (July 2023). ISSN: 0730-0301. DOI: 10.1145/3592430. URL: https://doi.org/10.1145/3592430.

[67] Y. Shi, P. Wang, J. Ye, L. Mai, K. Li, and X. Yang. "MVDream: Multi-view Diffusion for 3D Generation". In: *The Twelfth International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=FUgrjq2pbB.

[68] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. "Deep Unsupervised Learning using Nonequilibrium Thermodynamics". In: *Proceedings of the 32nd International Conference on Machine Learning*. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 2256–2265.

[69] J. Song, C. Meng, and S. Ermon. "Denoising Diffusion Implicit Models". In: *International Conference on Learning Representations*. 2021.

[70] Y. Song and S. Ermon. *Generative Modeling by Estimating Gradients of the Data Distribution*. 2020. arXiv: 1907.05600 [cs.LG].

[71] J. Tang, J. Ren, H. Zhou, Z. Liu, and G. Zeng. "DreamGaussian: Generative Gaussian Splatting for Efficient 3D Content Creation". In: *arXiv preprint arXiv:2309.16653* (2023).

[72] J. Tang, H. Zhou, X. Chen, T. Hu, E. Ding, J. Wang, and G. Zeng. "Delicate Textured Mesh Recovery from NeRF via Adaptive Surface Refinement". In: *arXiv preprint arXiv:2303.02091* (2022).

[73] D. Tochilkin, D. Pankratz, Z. Liu, Z. Huang, A. Letts, Y. Li, D. Liang, C. Laforte, V. Jampani, and Y.-P. Cao. *TripoSR: Fast 3D Object Reconstruction from a Single Image*. 2024. arXiv: 2403.02151 [cs.CV]. URL: https://arxiv.org/abs/2403.02151.

[74] M. Turk and A. Pentland. "Face recognition using eigenfaces". In: *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 1991, pp. 586–591. DOI: 10.1109/CVPR.1991.139758.

[75] Z. Wang, C. Lu, Y. Wang, F. Bao, C. Li, H. Su, and J. Zhu. *ProlificDreamer: High-Fidelity and Diverse Text-to-3D Generation with Variational Score Distillation*. 2023. arXiv: 2305.16213 [cs.LG]. URL: https://arxiv.org/abs/2305.16213.

[76] Wikipedia. *Volume of an n-ball — Wikipedia, The Free Encyclopedia*. http://en.wikipedia.org/w/index.php?title=Volume%20of%20an%20n-ball&oldid=1216556293. [Online; accessed 20-August-2024]. 2024.

[77] L. Zhang, Z. Wang, Q. Zhang, Q. Qiu, A. Pang, H. Jiang, W. Yang, L. Xu, and J. Yu. *CLAY: A Controllable Large-scale Generative Model for Creating High-quality 3D Assets*. 2024. arXiv: 2406.13897 [cs.CV]. URL: https://arxiv.org/abs/2406.13897.

[78] L. Zhang, A. Rao, and M. Agrawala. *Adding Conditional Control to Text-to-Image Diffusion Models*. 2023.

[79] S. Zuffi, A. Kanazawa, D. Jacobs, and M. J. Black. "3D Menagerie: Modeling the 3D Shape and Pose of Animals". In: *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. July 2017.