

Tetrahedron Splatting for 3D Generation

Chun Gu¹ Zeyu Yang¹ Zijie Pan¹ Xiatian Zhu² Li Zhang^{1*}

¹School of Data Science, Fudan University ²University of Surrey

<https://fudan-zvg.github.io/tet-splatting>



Figure 1: 3D assets generated by our proposed *TeT-Splatting*.

Abstract

3D representation is essential to the significant advance of 3D generation with 2D diffusion priors. As a flexible representation, NeRF has been first adopted for 3D representation. With density-based volumetric rendering, it however suffers both intensive computational overhead and inaccurate mesh extraction. Using a signed distance field and Marching Tetrahedra, DMTet allows for precise mesh extraction and real-time rendering but is limited in handling large topological changes in meshes, leading to optimization challenges. Alternatively, 3D Gaussian Splatting (3DGS) is favored in both training and rendering efficiency while falling short in mesh extraction. In this work, we introduce a novel 3D representation, Tetrahedron Splatting (*TeT-Splatting*), that supports easy convergence during optimization, precise mesh extraction, and real-time rendering *simultaneously*. This is achieved by integrating surface-based volumetric rendering within a structured tetrahedral grid while preserving the desired ability of precise mesh extraction, and a tile-based differentiable tetrahedron rasterizer. Furthermore, we incorporate eikonal and normal consistency regularization terms for the signed distance field to improve

*Li Zhang (lizhangfd@fudan.edu.cn) is the corresponding author.

generation quality and stability. Critically, our representation can be trained without mesh extraction, making the optimization process easier to converge. Our *TeT-Splatting* can be readily integrated in existing 3D generation pipelines, along with polygonal mesh for texture optimization. Extensive experiments show that our *TeT-Splatting* strikes a superior tradeoff among convergence speed, render efficiency, and mesh quality as compared to previous alternatives under varying 3D generation settings.

1 Introduction

Table 1: Comparison of different representations for 3D generation.

Representation	NeRF [28]	3DGS [13]	DMTet [40]	TeT-Splatting (Ours)
Precise mesh extraction			✓	✓
Easy convergence	✓	✓		✓
Real-time rendering		✓	✓	✓
Representative method	<i>DreamFusion</i> [32], <i>Magic3D</i> [18]	<i>DreamGaussian</i> [46], <i>GSGEN</i> [5]	<i>Fantasia3D</i> [3], <i>RichDreamer</i> [34]	<i>Ours</i>

Automatic 3D content generation is revolutionizing fields such as virtual reality, augmented reality, video games, and industrial design. This technology can significantly enhance user experiences and streamline creative processes for reducing time demands and simplifying the complexities associated with creating high-quality 3D assets.

3D representations (*e.g.*, Neural Radiance Field [28] (NeRF)) play an essential role in recent advancements in 3D generation, along with the Score Distillation Sampling (SDS) technique objective [32] for exploiting off-the-shelf 2D diffusion models [11, 37, 38, 1]. Although serving as a pioneer representation, NeRF is significantly limited due to its intensive computational demands, particularly when paired with high-resolution 2D diffusion models. Moreover, its density-based volumetric rendering struggles with accurate mesh extraction, which is crucial for practical applications.

By utilizing a signed distance field and Marching Tetrahedra for differentiable mesh extraction, DMTet [40] enables efficient high-resolution rendering and precise mesh extraction, overcoming the limitations of the NeRF approach. In cases, it becomes a favored choice [18, 33, 3]. However, DMTet is limited in its ability to handle large topological changes in meshes, as it can only backpropagate to the zero-level set of the signed distance field, constraining its geometry convergence during optimization. As a workaround, a two-stage 3D generation pipeline has been adopted that initially utilizes NeRF for rapid geometry convergence and then transitions to DMTet for detailed refinement [18]. However, transitioning from NeRF to DMTet often results in a degradation of quality, as the strengths of each representation are not fully leveraged throughout the entire optimization process.

Alternatively, recent methods have introduced 3D Gaussian Splatting [13] (3DGS) into the optimization process, significantly enhancing efficiency. For example, DreamGaussian [46] utilizes 3DGS but acknowledges that meshes directly generated from 3DGS can be blurry, and the mesh extraction process often results in unsatisfactory surfaces with visible holes [45]. Moreover, the text-to-3D process with 3DGS suffers from instability due to its unstructured nature and the densification process.

In this work, we introduce *TeT-Splatting*, a novel all-round 3D representation that integrates surface-based volumetric rendering into the tetrahedral grid, while preserving precise mesh extraction through Marching Tetrahedra. It supports easy convergence during optimization, precise mesh extraction, and real-time rendering *simultaneously*, enabling high-fidelity 3D generation effectively (Figure 1). Drawing inspiration from 3DGS [13], we design a tile-based fast differentiable rasterizer for real-time rendering, efficiently handling the alpha-blending of projected 2D splats from 3D tetrahedra. These splats are blended based on opacity values derived from the signed distance field within each tetrahedron as in NeuS [49]. To further increase efficiency, we include a pre-filtering process to remove nearly transparent tetrahedra, reducing the number of tetrahedra necessary for splatting. Moreover, we introduce eikonal and normal consistency regularization terms to refine the signed distance field, which helps stabilize the optimization process and prevents the common issue of debris in the optimization with DMTet. In Table 1 we compare the features of different 3D representations.

Our **contributions** are fourfold: (i) Introducing a novel 3D representation, *TeT-Splatting*, that integrates synergistically volumetric rendering into a tetrahedral grid; (ii) Designing a fast differentiable rasterizer for tetrahedra; (iii) Forming a generic two-stage 3D generation pipeline that initially leverages *TeT-Splatting* for geometry optimization, and then transitions it to polygonal mesh for texturing; (iv) Extensive evaluations demonstrating the superior tradeoff of our method among easy convergence, real-time rendering, and precise mesh extraction over alternative representations (InstantNGP [29], DMTet [40], and 3DGS [13]) under a variety of settings with different diffusion priors.

2 Related work

3D representation Since the introduction of Neural Radiance Field (NeRF) [28], NeRF has become a foundational technique in the field of 3D reconstruction. It employs volumetric rendering to enable 3D optimization with only 2D supervision. Despite its significance, NeRF faces major issues, such as slow rendering speeds and high memory usage. To address these problems, several research[43, 39, 29, 2, 13] have developed novel variants of the radiance field, focusing on faster training and rendering and using less computing resources. Diverging from the path of NeRF, DMTet [40] has introduced an approach based on Marching Tetrahedra and surface rendering by differentiable rasterization [14], offering much faster rendering speed. Recently, 3D Gaussian Splatting [13] (3DGS) has unified NeRF-like alpha-blending with tile-based rasterization, achieving high performance in both quality and rendering speed. In this paper, our proposed *TeT-Splatting* takes inspiration from the structured tetrahedral grid in DMTet [40] and incorporates tile-based rasterization from 3DGS [13], utilizing tetrahedra for splatting. *TeT-Splatting* achieves high converge and rendering speed while preserving precise mesh extraction through Marching Tetrahedra.

3D generation The data-driven 2D diffusion models [11, 37, 38, 1] have demonstrated unprecedented success in image generation. However, the transition to direct 3D generation [31, 12, 10, 24, 8, 58, 6, 52, 19, 15, 45] faces formidable challenges, as this research line often fails to generate high-quality 3D assets limited by the lack of training data. To circumvent these issues, some works [20, 41, 23, 42, 21, 25, 50] train 2D diffusion models to make them have 3D awareness. However, discrete and sparse 2D images still cannot offer sufficient 3D information. In this context, DreamFusion [32] first introduced score distillation sampling (SDS) loss to leverage 2D diffusion priors for 3D generation. Subsequent studies [47, 57, 59, 54, 51, 17, 26, 44, 48] have aimed to improve the SDS loss, enhancing both the fidelity and stability of 3D generation. Moreover, several efforts [55, 5, 56, 4, 16, 42, 34, 22] have been made to improve the quality and multi-view consistency of 3D models by integrating a wider array of diffusion priors. Despite these advancements, some methods are hindered by the significant computational demands due to the usage of NeRF [28], which limits the effective use of high-resolution diffusion priors. Additionally, other mesh-based models [3, 16, 34] encounter issues with instability and slow convergence due to the nature of surface rendering. By contrast, our *TeT-Splatting* facilitates the use of high-resolution diffusion priors and ensures efficient updates, thanks to its volumetric rendering and tile-based differentiable rasterizer.

3 TeT-Splatting

3.1 Deformable tetrahedral grid

In this section, we will start with an introduction to the deformable tetrahedral grid, which is the geometric primitive for the proposed representation. The deformable tetrahedral grid is first employed in DefTet [7] and then extended in DMTet [40] to approximate the implicit surface by assigning each vertex an SDF value. Specifically, this structure considers a tetrahedral mesh composed of N vertices and K tetrahedra, denoted as (V_T, T) , where $V_T = \{v_n | n \in 1, \dots, N\}$ signifies the positions of vertices, and $T = \{t_k | k \in 1, \dots, K\}$, with each t_k representing the indices (a_k, b_k, c_k, d_k) of four vertices $(v_{a_k}, v_{b_k}, v_{c_k}, v_{d_k})$ that form a tetrahedron. Utilizing the SDF value associated with each vertex v_n , denoted by f_n , a signed distance field is established by interpolating the SDF values within each tetrahedron. DMTet [40] has developed a method for mesh extraction from the tetrahedral grid by assigning one or two triangles to each tetrahedron that intersects the zero-level set of the signed distance field, known as Marching Tetrahedra (MT). Employing a differentiable triangular rasterizer, it attains a remarkable rendering speed while maintaining minimal memory consumption.

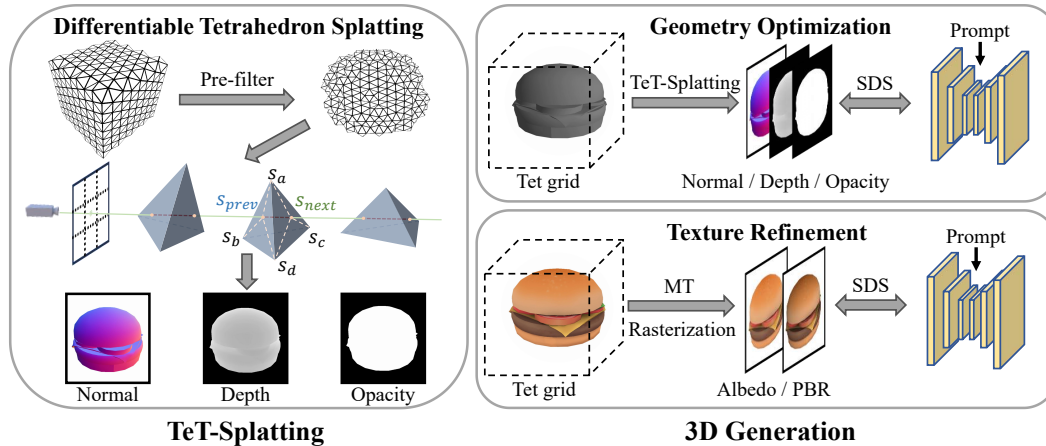


Figure 2: **Left: An overview of *TeT-Splatting*.** To produce the final renderings, we first pre-filter and remove nearly transparent tetrahedra, then project the remaining ones into 2D splats. These are blended based on opacity values derived from the SDF values at specific pixel intersections. **Right: *TeT-Splatting* for 3D generation.** We employ *TeT-Splatting* in the initial stage of the 3D generation pipeline and subsequently transition it to polygonal mesh for texture optimization.

However, a particular limitation of the MT is that only the parameters associated with tetrahedra intersecting the zero-level set of the signed distance field can be updated during optimization. This restriction poses challenges in managing large topological changes and often causes the optimization to stuck in the undesired shape in the early stage. In contrast, NeRF is less affected by such instability thanks to its volumetric nature. Many prior works [18, 33] have employed NeRF in 3D generation. These works typically adopt a two-stage pipeline that starts with the volumetric representation to swiftly achieve a coarse model with low-resolution diffusion priors and then transitions to a polygonal mesh for further refinement with high-resolution diffusion priors. However, these approaches are often hindered by the slow optimization and inaccurate geometry brought by volume rendering. The inaccurate geometry would lead to obvious degradation after mesh extraction.

3.2 Differentiable tetrahedron splatting

In this work, we present a unified representation that combines the precise mesh extraction via the tetrahedral grid and the efficient optimization of volumetric rendering. Inspired by 3D Gaussian Splatting [13] (3DGS), we also integrate the tile-based rasterizer into our framework to facilitate real-time rendering. 3DGS enhances rendering efficiency through rasterization and ensures efficient optimization via alpha-blending by projecting 3D Gaussians to 2D splats followed by fast alpha-blending. However, 3DGS relies on unstructured 3D Gaussians as rendering primitives, necessitating carefully designed densification processes and learning rates to manage the highly noisy SDS loss. In contrast, the tetrahedral grid is structured while its vertices can only deform in a local region and are connected with neighbors to form tetrahedra. We explore treating tetrahedron as rendering primitive of the splatting process to perform alpha-blending. Moreover, we can directly extract polygonal mesh through Marching Tetrahedra from the tetrahedral grid, while the mesh extracted [46] from 3DGS may result in an unsatisfactory surface with visible holes.

Next, we will elaborate on how we realize differentiable tetrahedron splatting through alpha-blending. Consider a pixel on the image plane, along with its corresponding ray in 3D space. To perform alpha-blending, we need first determine the intersected tetrahedra between the ray and the tetrahedral grid. For a single tetrahedron t with vertices (v_a, v_b, v_c, v_d) and SDF values (f_a, f_b, f_c, f_d) , we can project the vertices onto the image plane, resulting in four overlapped triangles that form a 2D tetrahedron splat. Intersection with the tetrahedron t is equal to the intersection with the four triangles. The position and SDF value of an intersection point can be calculated using barycentric coordinates (see Appendix A for details). Different from 3DGS [13], we consider the opacity of tetrahedra instead of Gaussians. Note that a ray can only have two intersection points with a tetrahedron, we denote their SDF values as f_{prev} and f_{next} in depth order. Then the opacity of the tetrahedron t can be derived

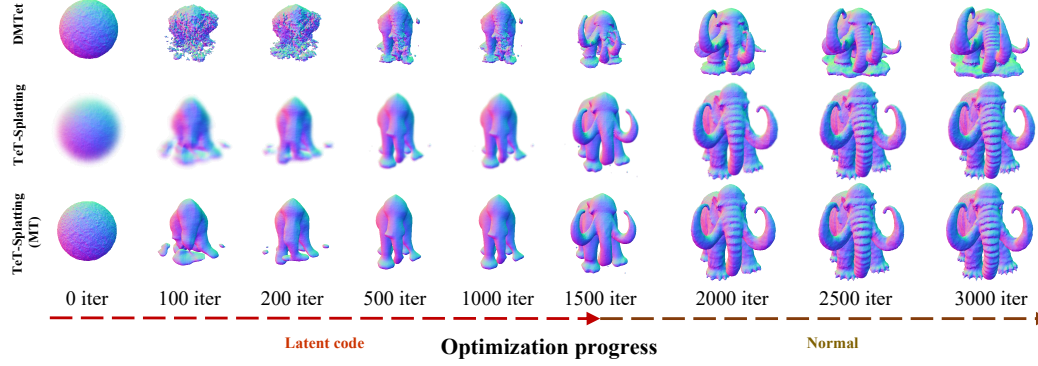


Figure 3: **Normal map comparison during optimization of 3D generation.** We utilize DMTet and *TeT-Splatting* as 3D representations in the geometry modeling stage of the RichDreamer [34]. The first two rows show normal maps obtained from DMTet and *TeT-Splatting* during optimization. *TeT-Splatting* achieves more stable and smooth optimization, while DMTet becomes fragmented initially and gets stuck in an undesirable shape. The third row shows the normal maps of meshes extracted from the signed distance field of *TeT-Splatting* via Marching Tetrahedra [40] (MT). As optimization progresses, *TeT-Splatting*’s behavior aligns with rendering through MT.

in NeuS [49] manner:

$$\alpha = \max \left(\frac{\Phi_s(f_{\text{prev}}) - \Phi_s(f_{\text{next}})}{\Phi_s(f_{\text{prev}})}, 0 \right), \quad (1)$$

where $\Phi_s(x) = (1 + e^{-sx})^{-1}$, and the s value controls the steepness of the conversion. Following Voxurf [53], we update s manually for each iteration i : $s = i/s_{\text{ratio}} + s_{\text{start}}$. The final normal map $\mathcal{N}$, depth map $\mathcal{D}$ and opacity map $\mathcal{O}$ are derived by alpha-blending N sequentially ordered tetrahedra from front to back:

$$\{\mathcal{N}, \mathcal{D}, \mathcal{O}\} = \sum_{i \in N} T_i \alpha_i \{\mathbf{n}_i, z_i, 1\}, \quad T_i = \prod_{j=1}^{i-1} (1 - \alpha_j), \quad (2)$$

where $\mathbf{n}$ denotes the per-tetrahedron normal and z_i denotes the average depths of four vertices.

Pre-filtering To conserve computational resources, the tetrahedra with low opacity will be filtered. Depending on different intersection points, the opacity of a tetrahedron can take different values. We can establish the upper bound of the opacity, denoted as α_{max} , by replacing s_{prev} and s_{next} in Eq. 1 with the maximum and minimum SDF values of four vertices. Tetrahedra with α_{max} less than a predefined threshold $T_f = \frac{1}{255}$ are filtered to ensure that only tetrahedra with significant enough contribution to the alpha-blending are included in the subsequent splatting process.

Per-tetrahedron normal As discussed in Section 3.1, the tetrahedral grid establishes a signed distance field by interpolating the SDF values within each tetrahedron. This interpolation is a linear combination of the SDF values of four vertices. Correspondingly, the barycentric coordinates of an arbitrary point with respect to the four vertices of the tetrahedron exhibit a linear correlation with its spatial position. This ensures that the gradient $\mathbf{g}$ of the SDF within the tetrahedron results in a constant vector (see Appendix A for details). The normal vector $\mathbf{n}$ of the tetrahedron is thus obtained by normalizing this gradient.

Relationship between DMTet and TeT-Splatting During optimization, DMTet employs Marching Tetrahedra to extract polygonal mesh from the tetrahedral grid and subsequently renders through triangular rasterization [14]. Consequently, only a limited number of tetrahedra are involved in each single rendering process. In contrast, *TeT-Splatting* employs volumetric rendering, which allows all visible tetrahedra within the view frustum that have sufficient weight in alpha-blending to contribute to the final renderings. Moreover, the rendering process in *TeT-Splatting* is fully differentiable, enabling a single optimization step to influence a significantly larger number of parameters compared to DMTet. Figure 3 presents a comparative analysis of convergence speeds between DMTet and *TeT-Splatting* within the same Text-to-3D pipeline. As observed, *TeT-Splatting* achieves rapid convergence, whereas DMTet exhibits slower topological changes and gets stuck in an undesirable shape. Furthermore, as the inverse standard deviation s in Eq. 1 increases, the curve of $\Phi_s(x)$ becomes steeper, causing

α to approach 1 under conditions that $s_{prev} > 0$ and $s_{next} < 0$ and to approach 0 otherwise. This behavior (Figure 3) aligns with the rendering process of DMTet [40] through Marching Tetrahedra, where only the tetrahedra intersecting the zero-level set of the signed distance field are visible.

3.3 Fast differentiable rasterizer for tetrahedra

We implement a tile-based differentiable rasterizer for tetrahedra with custom CUDA kernel building upon the framework of 3DGS [13]. Similarly, we begin by dividing the screen into tiles and culling tetrahedra that do not overlap with the view frustum. We then replicate the tetrahedra based on the number of tiles they overlap and sort them by their tile ID and the average depth of each tetrahedron's vertices, using a fast GPU radix sort [27]. Note that the per-tile sorting in 3DGS is not equivalent to per-pixel ordering. Differently, we maintain a short resorting window [36] of size N_w for each pixel to re-sort the primitives based on the results of per-tile sorting using the insertion sort. Due to the structured nature of the tetrahedral grid, we find that the sorting error is almost eliminated with a window size of 5 under a grid resolution of 256. The operations after re-sorting for alpha-blending are the same as in 3DGS, except for the computation of α , which we have already given in Eq. 1.

4 3D generation with TeT-Splatting

In this section, we introduce our 3D generation pipeline and discuss various settings, aiming at validating the effectiveness of *TeT-Splatting* in 3D generation. As shown in Figure 2, our pipeline is divided into two stages: first get a detailed geometry with *TeT-Splatting* and then transition to polygonal mesh through Marching Tetrahedra [40] for texture optimization. We begin by describing our overall 3D model (Section 4.1), and then detail the regularizations (Section 4.2) and diffusion priors (Section 4.3) used in our experiments.

4.1 3D modeling

Geometry stage We employ a hash grid [29] Φ_g with parameter Θ_g to encode the signed distance field and deformation which allows each vertex in a tetrahedral grid to deform in a certain range. Φ_g is initialized to a spherical shape. Given a randomly sampled camera, our tetrahedron rasterizer produces renderings of the normal map, depth map, and opacity map.

Texture stage Given the well-optimized signed distance field from the geometry stage, we convert it into a polygonal mesh through Marching Tetrahedra. To texture the polygonal mesh, we employ the physically based rendering (PBR) pipeline proposed by Nvdiffrac [30]. Please refer to [30, 9, 34] for details. We use another hash grid Φ_t with parameter Θ_t to encode the spatially varying materials of the surface: albedo, roughness, metallic, and bump. Finally, given a specific environment lighting and a randomly sampled camera, we can obtain the renderings of the albedo map and PBR map.

4.2 Regularization

Eikonal loss To ensure a proper signed distance field, we employ an eikonal term that regularizes the SDF gradient $\mathbf{g}$ in each tetrahedron: $\mathcal{L}_{\text{eik}} = \sum_k (\|\mathbf{g}_k\|_2 - 1)^2$.

Normal consistency loss Inspired by the normal consistency loss for triangle meshes, we adapt this approach to tetrahedra. While we have designed a per-tetrahedron normal, we project these tetrahedral normals onto vertices and enhance the consistency of the signed distance field by regularizing the cosine similarity between normals of adjacent vertices connected by edges: $\mathcal{L}_{\text{nc}} = \sum_i (1 - \cos(\mathbf{n}_{e_{i1}}, \mathbf{n}_{e_{i2}}))$, where e_{i1} and e_{i2} represent the vertices forming edge e_i .

4.3 Diffusion priors

To validate the capability of *TeT-Splatting* for 3D generation, we employ two types of diffusion priors: the vanilla RGB-based diffusion priors and the rich diffusion priors proposed in RichDreamer [34].

Vanilla RGB-based diffusion priors Vanilla RGB-based diffusion models represent diffusion models that can generate RGB images from a given prompt. For both geometry and texture stages, we utilize SDS loss to leverage 2D diffusion priors from Stable Diffusion [37]:

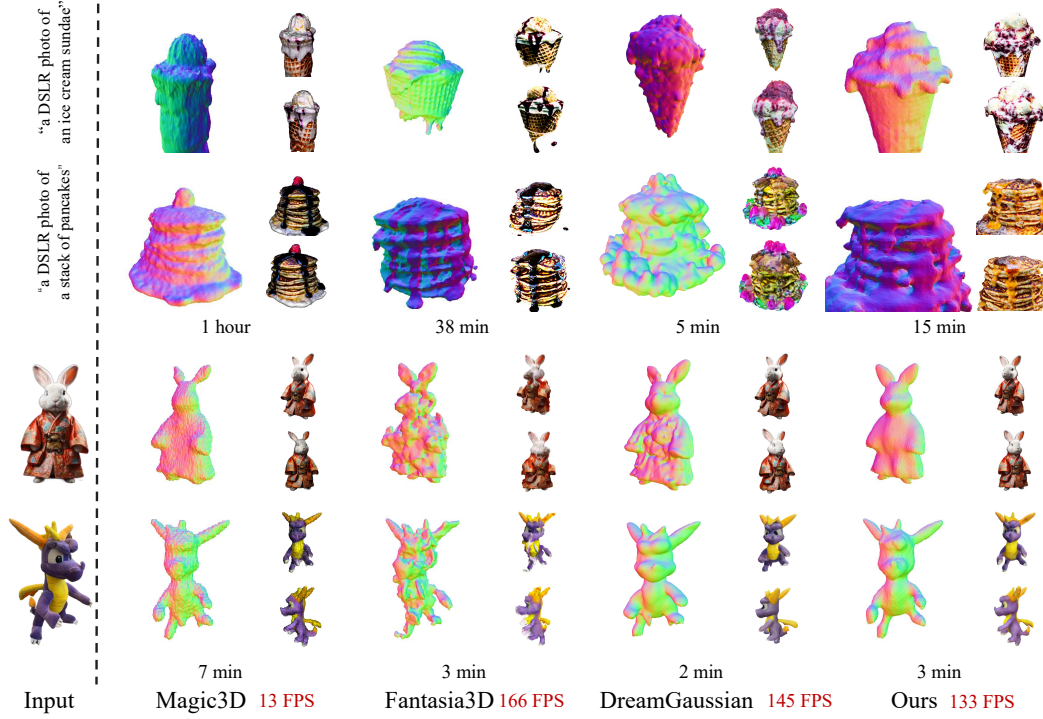


Figure 4: Qualitative comparison on 3D generation using vanilla RGB-based diffusion priors. We present visual comparisons of the rendered RGB maps and color maps from various 3D generation methods. The methods, arranged from left to right, are: Magic3D, Fantasia3D, DreamGaussian, and Ours. The comparison is conducted across two tasks, text-to-3D and image-to-3D, with results shown from top to bottom, respectively. Additionally, for each method, we provide the training time and the rendering speed (FPS) for the first stage of the process.

$\nabla_{\Theta} \mathcal{L}_{\text{SDS}} = \mathbb{E} [\omega(t)(\epsilon_{\phi}(\mathcal{I}; y, t) - \epsilon) \frac{\partial z}{\partial \mathcal{I}} \frac{\partial \mathcal{I}}{\partial \Theta}]$, where $\omega(t)$ is a weighting function, z denotes the VAE latent code, and $\epsilon_{\phi}(\mathcal{N}; y, t)$ represents the noise estimated by the UNet ϵ_{ϕ} .

Rich diffusion priors The sole use of vanilla diffusion priors often leads to issues such as multi-face Janus problem, domain gap between image diffusion model and normal map while using normal maps as input of diffusion models, and inaccuracies in material decomposition. To this end, we utilize the rich diffusion priors proposed in RichDreamer [34] to handle high-fidelity 3D generation. Specifically, for geometry optimization, we combine a vanilla Stable Diffusion with a Normal-Depth diffusion model, which generates multi-view normal and depth maps from a given text prompt, represented as: $\mathcal{L}_{\text{SDS}} = \mathcal{L}_{\text{SDS-Normal}}^{\text{SD}} + \mathcal{L}_{\text{SDS-ND}}^{\text{ND}}$. For texture optimization, we combine a vanilla SD with a Depth-conditioned Albedo diffusion model, capable of producing multi-view albedo maps from a given text prompt, represented as: $\mathcal{L}_{\text{SDS}} = \mathcal{L}_{\text{SDS-RGB}}^{\text{SD}} + \mathcal{L}_{\text{SDS-Albedo}}^{\text{Albedo}}$.

In summary, the final loss function for the geometry stage is defined as: $\mathcal{L}_{\text{geo}} = \mathcal{L}_{\text{SDS}} + \lambda_{\text{eik}} \mathcal{L}_{\text{eik}} + \lambda_{\text{nc}} \mathcal{L}_{\text{nc}}$. For the texture stage, the loss function simplifies to: $\mathcal{L}_{\text{tex}} = \mathcal{L}_{\text{SDS}}$.

5 Experiment

In this section, we assess the efficacy of *TeT-Splatting* across two distinct tasks: 3D generation employing vanilla RGB-based diffusion priors and text-to-3D with rich diffusion priors. In Section 5.1, a qualitative evaluation of 3D generation for both text-to-3D and image-to-3D modalities is conducted to demonstrate the superiority of *TeT-Splatting* relative to other representations. To substantiate *TeT-Splatting*'s proficiency in handling high-fidelity generations, we conduct experiments with advanced rich diffusion priors in Section 5.2. Section 5.3 encompasses a series of ablation studies aimed at validating the representation and pipeline. The details of implementation and experimental setting can be found in the Appendix B.

5.1 Results with vanilla RGB-based diffusion priors



Figure 5: Visualization of normal maps before and after mesh exportation. Note that the normal maps of DreamGaussian [46] are derived from its depth maps.

Focusing on illustrating the effectiveness of the proposed representation, we primarily compare to three competitors: Magic3D [18], Fantasia3D [3] and DreamGaussian [46]. All these competitors employ a two-stage optimization pipeline and leverage Stable Diffusion with SDS loss, but utilize three different representative 3D representations in their initial stages: Instant-NGP [29] (a fast version of NeRF), DMTet [40], and 3DGS [13], respectively. We adapt the diffusion priors of all methods to Stable Zero-1-to-3 [20] for a fair comparison in the image-to-3D task, adding an identical MSE alignment loss. The qualitative evaluations, as shown in Figure 4, illustrate our method's ability to generate more detailed and compact meshes in a relatively short time. In Figure 4, we also report the rendering speed (FPS) of the first stage, at a rendering resolution of 512x512. Although *TeT-Splatting* operates at a lower FPS compared to DMTet (Fantasia3D) and 3DGS (DreamGaussian), it still achieves real-time rendering. Importantly, this lower FPS does not adversely affect the overall generation process, as the primary bottleneck in generation speed lies with the diffusion model.

We also conduct a comparison of the mesh extraction with Magic3D and DreamGaussian, visualized in Figure 5. It reveals that the meshes extracted from Magic3D often do not faithfully replicate the geometries from the first stage due to the imprecise threshold for converting densities to SDF values, and the extracted mesh in DreamGaussian can result in unsatisfactory surfaces with visible holes. In contrast, our method maintains high quality with negligible degradation after mesh extraction.

5.2 Results with rich diffusion priors

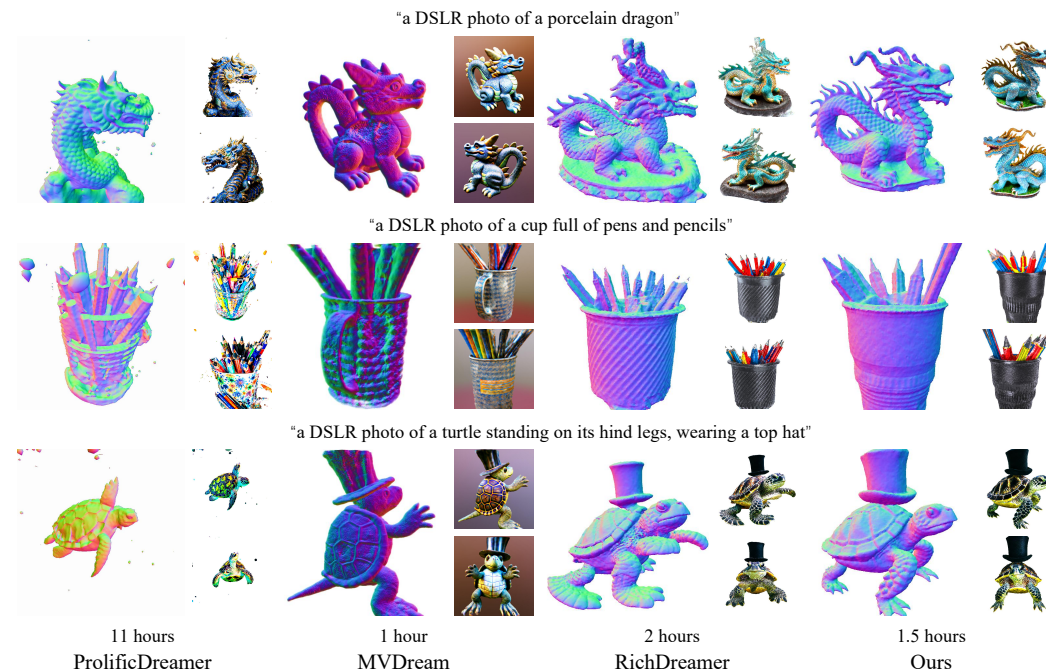


Figure 6: Qualitative comparison on Text-to-3D with rich diffusion priors. We also report the total training time of each method.

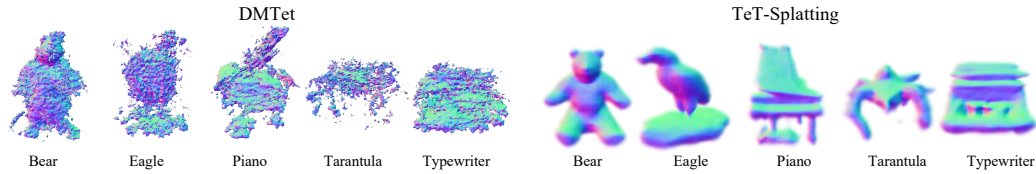


Figure 7: Normal map comparison between DMTet [40] and *TeT-Splatting* in the early training iterations.

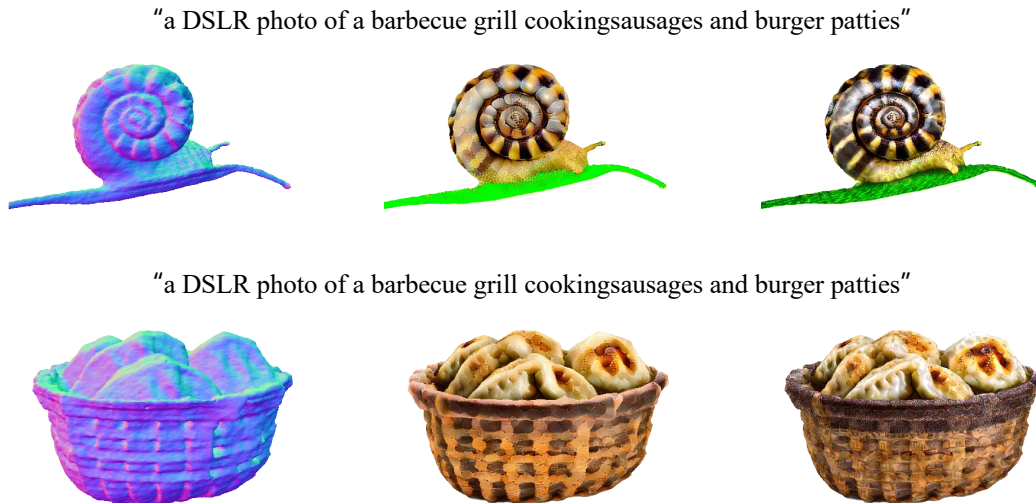


Figure 8: Visualization of the rendered normal, albedo, and PBR map from the generated 3D assets in the second stage.

Our approach is also compatible with state-of-the-art diffusion priors. In this part, we evaluate *TeT-Splatting* on the text-to-3D task, equipped with rich diffusion priors from RichDreamer [34]. A key distinction between our approach and RichDreamer is the use of *TeT-Splatting* as the 3D representation during the geometry stage. We evaluate our method against two SOTA competitors, ProlificDreamer [51] and RichDreamer [34]. As illustrated in Figure 6, our method is capable of handling high-fidelity 3D generation, achieving superior geometric quality with considerably reduced generation times compared to these competitors.

Additionally, we present visualizations of the normal maps from early training iterations in Figure 7. The results from RichDreamer are fragmented at the early iterations due to the use of DMTet, which may harm subsequent optimization and slow convergence. In contrast, *TeT-Splatting* demonstrates rapid and smooth convergence. We employ the same quantitative evaluation method as RichDreamer to assess the quality of geometry and texture.

In Table 2, we report the Geometry CLIP [35] score and Appearance CLIP score. Notably, RichDreamer’s prompt list, comprising 113 objects used for scoring, is not publicly available. Consequently, we calculate our scores using an alternative set of prompts (see Appendix B for details). Our method outperforms RichDreamer in terms of CLIP scores and significantly reduces the time required for geometry optimization (40 min vs 70 min).

In Figure 8, we present the decomposed albedo maps of generated 3D assets. Guided by the Depth-conditioned Albedo diffusion model, we achieve natural albedo maps.

5.3 Ablation

Under the settings of rich diffusion priors [34], we conduct ablation studies to evaluate our method.

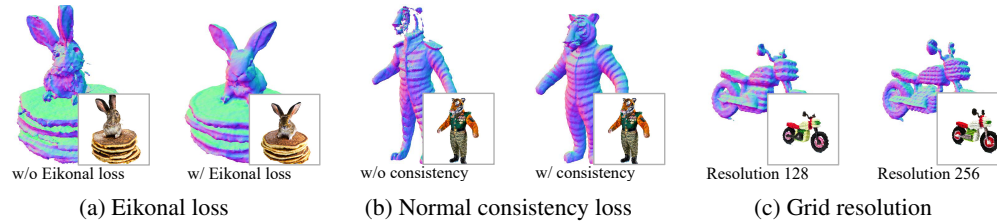


Figure 9: **Ablation studies** on eikonal loss, normal consistency loss and the resolution of the tetrahedral grid.

Table 2: **CLIP score comparison**. Results marked with “*” are taken from RichDreamer [34]. Since RichDreamer [34] did not release their prompt list (113 objects), we use our own prompt list (183 objects) for evaluation. See Appendix B for more details.

	Prolificdreamer [51]	MVDream [42]	RichDreamer [34]	RichDreamer [34]	Ours
Geometry CLIP score ↑	23.3818*	24.8003*	25.8820*	23.0143	23.1641
Appearance CLIP score ↑	31.8022*	28.7331*	31.7099*	29.2198	29.4197

Eikonal loss We assess the role of eikonal loss in 3D generation by comparing 3D assets generated with and without it, illustrated in Figure 9a. Models created without eikonal loss tend to develop into undesirable shapes. This issue arises because the SDF values rapidly reach extreme levels and get trapped in local minima when eikonal loss is not applied.

Normal consistency loss Additionally, we assess the importance of normal consistency loss. Figure 9b demonstrates that applying normal consistency loss results in more compact models. This loss can act as a smoothing prior that helps prevent the surface of the model from becoming fragmented.

Tetrahedral grid resolution We investigate the effects of tetrahedral grid resolution on model performance by conducting experiments at resolutions of 128 and 256. Higher resolution yields more detailed geometries, as shown in Figure 9c.

6 Limitations

TeT-Splatting struggles with modeling high-frequency features, such as texture, because it uses tetrahedra as rendering primitives, which limits the final output by the resolution of the tetrahedral grid. Therefore, we transition it to a polygonal mesh for enhanced texture optimization. The rendering speed of our implemented rasterizer, although operating in real-time, is slower than that of 3DGS. Additionally, using only a pre-filter operation might not fully leverage *TeT-Splatting*’s potential in rendering quality and speed. A similar densification process as in 3DGS could improve this, which we leave for future work.

7 Conclusion

In this study, we introduce Tetrahedron Splatting (*TeT-Splatting*), a novel all-round 3D representation that integrates volumetric rendering within a structured tetrahedral grid while preserving precise mesh extraction through Marching Tetrahedra. Equipped with newly designed tile-based fast differentiable tetrahedron rasterizer, *TeT-Splatting* achieves real-time rendering. As showcase, we integrate *TeT-Splatting* in common 3D generation pipeline with polygonal mesh for texture optimization. Extensive experiments under varying 3D generation settings demonstrate *TeT-Splatting*’s superiority in producing high-fidelity 3D content compared to other 3D representations.

Acknowledgments

This work was supported in part by National Natural Science Foundation of China (Grant No. 62106050 and 62376060) and Natural Science Foundation of Shanghai (Grant No. 22ZR1407500).

References

- [1] Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Qinsheng Zhang, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, et al. ediff-i: Text-to-image diffusion models with an ensemble of expert denoisers. *arXiv preprint*, 2022. 2, 3
- [2] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *ECCV*, 2022. 3
- [3] Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. In *ICCV*, 2023. 2, 3, 8, 15
- [4] Yang Chen, Yingwei Pan, Yehao Li, Ting Yao, and Tao Mei. Control3d: Towards controllable text-to-3d generation. In *ACM MM*, 2023. 3
- [5] Zilong Chen, Feng Wang, and Huaping Liu. Text-to-3d using gaussian splatting. In *CVPR*, 2024. 2, 3
- [6] Yen-Chi Cheng, Hsin-Ying Lee, Sergey Tuyakov, Alex Schwing, and Liangyan Gui. SDFusion: Multimodal 3d shape completion, reconstruction, and generation. In *CVPR*, 2023. 3
- [7] Jun Gao, Wenzheng Chen, Tommy Xiang, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Learning deformable tetrahedral meshes for 3d reconstruction. In *NeurIPS*, 2020. 3
- [8] Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojcic, and Sanja Fidler. Get3d: A generative model of high quality 3d textured shapes learned from images. In *NeurIPS*, 2022. 3
- [9] Yuan-Chen Guo, Ying-Tian Liu, Ruizhi Shao, Christian Laforte, Vikram Voleti, Guan Luo, Chia-Hao Chen, Zi-Xin Zou, Chen Wang, Yan-Pei Cao, and Song-Hai Zhang. threestudio: A unified framework for 3d content generation. <https://github.com/threestudio-project/threestudio>, 2023. 6, 15
- [10] Anchit Gupta, Wenhan Xiong, Yixin Nie, Ian Jones, and Barlas Oğuz. 3dgen: Triplane latent diffusion for textured mesh generation. *arXiv preprint*, 2023. 3
- [11] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020. 2, 3
- [12] Heewoo Jun and Alex Nichol. Shap-e: Generating conditional 3d implicit functions. *arXiv preprint*, 2023. 3
- [13] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM TOG*, 2023. 2, 3, 4, 6, 8, 15
- [14] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. *ACM TOG*, 2020. 3, 5
- [15] Ming Li, Pan Zhou, Jiawei Liu, Jussi Keppo, Min Lin, Shuicheng Yan, and Xiangyu Xu. Instant3d: Instant text-to-3d generation. *arXiv preprint*, 2023. 3
- [16] Weiyu Li, Rui Chen, Xuelin Chen, and Ping Tan. Sweetdreamer: Aligning geometric priors in 2d diffusion for consistent text-to-3d. In *ICLR*, 2024. 3
- [17] Yixun Liang, Xin Yang, Jiantao Lin, Haodong Li, Xiaogang Xu, and Yingcong Chen. Lucid-dreamer: Towards high-fidelity text-to-3d generation via interval score matching. In *CVPR*, 2024. 3
- [18] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation. In *CVPR*, 2023. 2, 4, 8, 15
- [19] Minghua Liu, Chao Xu, Haian Jin, Linghao Chen, Mukund Varma T, Zexiang Xu, and Hao Su. One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization. In *NeurIPS*, 2024. 3
- [20] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object. In *ICCV*, 2023. 3, 8
- [21] Yuan Liu, Cheng Lin, Zijiao Zeng, Xiaoxiao Long, Lingjie Liu, Taku Komura, and Wenping Wang. Syncdreamer: Learning to generate multiview-consistent images from a single-view image. In *CVPR*, 2024. 3

- [22] Zexiang Liu, Yangguang Li, Youtian Lin, Xin Yu, Sida Peng, Yan-Pei Cao, Xiaojuan Qi, Xiaoshui Huang, Ding Liang, and Wanli Ouyang. Unidream: Unifying diffusion priors for reliable text-to-3d generation. *arXiv preprint*, 2023. 3
- [23] Xiaoxiao Long, Yuan-Chen Guo, Cheng Lin, Yuan Liu, Zhiyang Dou, Lingjie Liu, Yuexin Ma, Song-Hai Zhang, Marc Habermann, Christian Theobalt, et al. Wonder3d: Single image to 3d using cross-domain diffusion. *arXiv preprint*, 2023. 3
- [24] Jonathan Lorraine, Kevin Xie, Xiaohui Zeng, Chen-Hsuan Lin, Towaki Takikawa, Nicholas Sharp, Tsung-Yi Lin, Ming-Yu Liu, Sanja Fidler, and James Lucas. Att3d: Amortized text-to-3d object synthesis. In *ICCV*, 2023. 3
- [25] Yuanxun Lu, Jingyang Zhang, Shiwei Li, Tian Fang, David McKinnon, Yanghai Tsin, Long Quan, Xun Cao, and Yao Yao. Direct2. 5: Diverse text-to-3d generation via multi-view 2.5 d diffusion. *arXiv preprint*, 2023. 3
- [26] Yiwei Ma, Yijun Fan, Jiayi Ji, Haowei Wang, Xiaoshuai Sun, Guannan Jiang, Annan Shu, and Rongrong Ji. X-dreamer: Creating high-quality 3d content by bridging the domain gap between text-to-2d and text-to-3d generation. *arXiv preprint*, 2023. 3
- [27] Duane G Merrill and Andrew S Grimshaw. Revisiting sorting for gpgpu stream architectures. In *ICPAC*, 2010. 6
- [28] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 2, 3
- [29] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM TOG*, 2022. 3, 6, 8
- [30] Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. Extracting triangular 3d models, materials, and lighting from images. In *CVPR*, 2022. 6
- [31] Alex Nichol, Heewoo Jun, Pratul Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generating 3d point clouds from complex prompts. *arXiv preprint*, 2022. 3
- [32] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *ICLR*, 2023. 2, 3
- [33] Guocheng Qian, Jinjie Mai, Abdullah Hamdi, Jian Ren, Aliaksandr Siarohin, Bing Li, Hsin-Ying Lee, Ivan Skorokhodov, Peter Wonka, Sergey Tulyakov, et al. Magic123: One image to high-quality 3d object generation using both 2d and 3d diffusion priors. In *ICLR*, 2023. 2, 4
- [34] Lingteng Qiu, Guanying Chen, Xiaodong Gu, Qi zuo, Mutian Xu, Yushuang Wu, Weihao Yuan, Zilong Dong, Liefeng Bo, and Xiaoguang Han. Richdreamer: A generalizable normal-depth diffusion model for detail richness in text-to-3d. *arXiv preprint*, 2023. 2, 3, 5, 6, 7, 9, 10, 16
- [35] Alec Radford, Jong Wook Kim, Chris Hallacy, A. Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 9, 16
- [36] Lukas Radl, Michael Steiner, Mathias Parger, Alexander Weinrauch, Bernhard Kerbl, and Markus Steinberger. Stopthepop: Sorted gaussian splatting for view-consistent real-time rendering. *arXiv preprint*, 2024. 6
- [37] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 2, 3, 6
- [38] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. In *NeurIPS*, 2022. 2, 3
- [39] Sara Fridovich-Keil and Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022. 3
- [40] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. In *NeurIPS*, 2021. 2, 3, 5, 6, 8, 9, 16

- [41] Ruoxi Shi, Hansheng Chen, Zhuoyang Zhang, Minghua Liu, Chao Xu, Xinyue Wei, Linghao Chen, Chong Zeng, and Hao Su. Zero123++: a single image to consistent multi-view diffusion base model. *arXiv preprint*, 2023. 3
- [42] Yichun Shi, Peng Wang, Jianglong Ye, Long Mai, Kejie Li, and Xiao Yang. Mvdream: Multi-view diffusion for 3d generation. In *ICLR*, 2024. 3, 10
- [43] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, 2022. 3
- [44] Boshi Tang, Jianan Wang, Zhiyong Wu, and Lei Zhang. Stable score distillation for high-quality 3d generation. *arXiv preprint*, 2023. 3
- [45] Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu. Lgm: Large multi-view gaussian model for high-resolution 3d content creation. *arXiv preprint*, 2024. 2, 3
- [46] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. In *ICLR*, 2024. 2, 4, 8
- [47] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A Yeh, and Greg Shakhnarovich. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. In *CVPR*, 2023. 3
- [48] Peihao Wang, Zhiwen Fan, Dejia Xu, Dilin Wang, Sreyas Mohan, Forrest Iandola, Rakesh Ranjan, Yilei Li, Qiang Liu, Zhangyang Wang, et al. Steindreamer: Variance reduction for text-to-3d score distillation via stein identity. *arXiv preprint*, 2023. 3
- [49] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint*, 2021. 2, 5
- [50] Peng Wang and Yichun Shi. Imagedream: Image-prompt multi-view diffusion for 3d generation. *arXiv preprint*, 2023. 3
- [51] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Pro-lificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. In *NeurIPS*, 2024. 3, 9, 10
- [52] Chao-Yuan Wu, Justin Johnson, Jitendra Malik, Christoph Feichtenhofer, and Georgia Gkioxari. Multiview compressive coding for 3D reconstruction. In *CVPR*, 2023. 3
- [53] Tong Wu, Jiaqi Wang, Xingang Pan, Xudong Xu, Christian Theobalt, Ziwei Liu, and Dahua Lin. Voxurf: Voxel-based efficient and accurate neural surface reconstruction. In *ICLR*, 2023. 5
- [54] Zike Wu, Pan Zhou, Xuanyu Yi, Xiaoding Yuan, and Hanwang Zhang. Consistent3d: Towards consistent high-fidelity text-to-3d generation with deterministic sampling prior. In *ICLR*, 2024. 3
- [55] Jiale Xu, Xintao Wang, Weihao Cheng, Yan-Pei Cao, Ying Shan, Xiaohu Qie, and Shenghua Gao. Dream3d: Zero-shot text-to-3d synthesis using 3d shape prior and text-to-image diffusion models. In *CVPR*, 2023. 3
- [56] Taoran Yi, Jiemin Fang, Junjie Wang, Guanjun Wu, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Qi Tian, and Xinggang Wang. Gaussiandreamer: Fast generation from text to 3d gaussians by bridging 2d and 3d diffusion models. In *CVPR*, 2024. 3
- [57] Xin Yu, Yuan-Chen Guo, Yangguang Li, Ding Liang, Song-Hai Zhang, and Xiaojuan Qi. Text-to-3d with classifier score distillation. *arXiv preprint*, 2023. 3
- [58] Xiaohui Zeng, Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, and Karsten Kreis. Lion: Latent point diffusion models for 3d shape generation. In *NeurIPS*, 2022. 3
- [59] Junzhe Zhu, Peiye Zhuang, and Sanmi Koyejo. Hifa: High-fidelity text-to-3d generation with advanced diffusion guidance. In *ICLR*, 2023. 3

A More implementation details of TeT-Splatting

In this section, we provide additional implementation details about the tetrahedron splatting process.

A.1 Barycentric coordinates

Recall that projecting a single tetrahedron, encompassing four vertices $\mathbf{v}_a, \mathbf{v}_b, \mathbf{v}_c, \mathbf{v}_d \in \mathbb{R}^3$ along with their SDF values $f_a, f_b, f_c, f_d \in \mathbb{R}$, onto a 2D plane results in an array of 2D vectors $\mathbf{v}'_a, \mathbf{v}'_b, \mathbf{v}'_c, \mathbf{v}'_d \in \mathbb{R}^2$ representing the coordinates of these vertices on the image plane.

Given a pixel $\mathbf{p}$ with coordinate $\mathbf{v}'_p \in \mathbb{R}^2$, its barycentric coordinates $u', v' \in \mathbb{R}$ with respect to a triangle $(\mathbf{v}'_a, \mathbf{v}'_b, \mathbf{v}'_c)$ satisfy $\mathbf{v}'_p = (1 - u' - v')\mathbf{v}'_a + u'\mathbf{v}'_b + v'\mathbf{v}'_c$. We can apply the following equations to derive the barycentric coordinates u', v' :

$$\begin{bmatrix} \mathbf{v}'_a & \mathbf{v}'_b & \mathbf{v}'_c \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 - u' - v' \\ u' \\ v' \end{bmatrix} = \begin{bmatrix} \mathbf{v}'_p \\ 1 \end{bmatrix}, \quad (3)$$

$$\Rightarrow \begin{bmatrix} 1 - u' - v' \\ u' \\ v' \end{bmatrix} = \begin{bmatrix} \mathbf{v}'_a & \mathbf{v}'_b & \mathbf{v}'_c \\ 1 & 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{v}'_p \\ 1 \end{bmatrix}. \quad (4)$$

If $u', v' \in [0, 1]$, the pixel $\mathbf{p}$ is considered inside the triangle.

A.2 Perspective correction

The calculated barycentric coordinates u', v' are based on 2D projections and require adjustment to reflect the original 3D spatial relationships accurately. This adjustment, known as perspective correction, is necessary because 3D depth information is not preserved in the 2D projection. We perform this correction using:

$$u = \frac{\frac{u'}{z_b}}{\frac{(1-u'-v')}{z_a} + \frac{u'}{z_b} + \frac{v'}{z_c}}, \quad v = \frac{\frac{v'}{z_c}}{\frac{(1-u'-v')}{z_a} + \frac{u'}{z_b} + \frac{v'}{z_c}}, \quad (5)$$

where z_* denotes the depth of each vertex. Subsequently, the SDF value and depth of the 3D position corresponding to this pixel are interpolated:

$$f_p = (1 - u - v)f_a + uf_b + vf_c, \quad z_p = \frac{1}{\frac{(1-u'-v')}{z_a} + \frac{u'}{z_b} + \frac{v'}{z_c}}. \quad (6)$$

A.3 Gradient of the SDF value inside a tetrahedron

Consider an arbitrary 3D point $\mathbf{v}_q$ with SDF value f_q inside the tetrahedron $(\mathbf{v}_a, \mathbf{v}_b, \mathbf{v}_c, \mathbf{v}_d)$. We establish the SDF value f_q using the 3D barycentric coordinates u, v, w : $f_q = (1 - u - v - w)f_a + uf_b + vf_c + wf_d$. The derivation of u, v, w is similar to 2D case in Section A.1, i.e.,

$$\begin{bmatrix} \mathbf{v}_a & \mathbf{v}_b & \mathbf{v}_c & \mathbf{v}_d \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 - u - v - w \\ u \\ v \\ w \end{bmatrix} = \begin{bmatrix} \mathbf{v}_q \\ 1 \end{bmatrix}, \quad (7)$$

$$\Rightarrow \begin{bmatrix} 1 - u - v - w \\ u \\ v \\ w \end{bmatrix} = \begin{bmatrix} \mathbf{v}_a & \mathbf{v}_b & \mathbf{v}_c & \mathbf{v}_d \\ 1 & 1 & 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{v}_q \\ 1 \end{bmatrix}. \quad (8)$$

Therefore, the formula of f_q can be expressed by:

$$f_q = [f_a \quad f_b \quad f_c \quad f_d] \begin{bmatrix} 1 - u - v - w \\ u \\ v \\ w \end{bmatrix} \quad (9)$$

$$= \begin{bmatrix} f_a & f_b & f_c & f_d \end{bmatrix} \begin{bmatrix} \mathbf{v}_a & \mathbf{v}_b & \mathbf{v}_c & \mathbf{v}_d \\ 1 & 1 & 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{v}_q \\ 1 \end{bmatrix}. \quad (10)$$

Moreover, the gradient of the SDF value at $\mathbf{v}_q$, denoted by $\mathbf{g}$, is straightforwardly derived by differentiating with respect to $\mathbf{v}_q$. This gradient is constant across the tetrahedron:

$$\begin{bmatrix} \mathbf{g} \\ * \end{bmatrix} = \begin{bmatrix} \mathbf{v}_a^\top & 1 \\ \mathbf{v}_b^\top & 1 \\ \mathbf{v}_c^\top & 1 \\ \mathbf{v}_d^\top & 1 \end{bmatrix}^{-1} \begin{bmatrix} f_a \\ f_b \\ f_c \\ f_d \end{bmatrix}. \quad (11)$$

This results in a constant vector within any tetrahedron, providing a consistent gradient that aids in precise mesh extractions and surface optimizations.

B More implementation details of 3D generation

In this section, we provide additional implementation details of 3D generation. Note that all experiments are conducted on one NVIDIA RTX A6000 GPU.

B.1 Geometry stage

Unlike 3DGS [13], we separate operations that are repeated across multiple images from different camera viewpoints within a single training iteration and shift them to the beginning of each iteration, including the inference of SDF values and deformations for each vertex from the hash grid, pre-filtering tetrahedra based on their α_{max} , and calculating the per-tetrahedron normal. These pre-processed results are then passed to the rasterizer for rendering a batch of images. Moreover, we implement a coarse-to-fine approach in the pre-filtering process: initially, we establish a tighter axis-aligned bounding box from the pre-filtered tetrahedral grid in the first round and then resize the tetrahedral grid based on this bounding box for a second round of pre-filtering, which enhances the precision of the geometry. For the schedule of the s value in Eq. 1, we set $s_{ratio} = 5$ and $s_{start} = 20$, which allows the curve of $\Phi_s(x)$ to be sufficiently steep at the final of optimization. Additionally, we set both λ_{eik} and λ_{nc} to 1000.

B.2 Evaluation with vanilla RGB-based diffusion priors

This part is implemented based on the threestudio codebase [9] using the settings of Fantasia3D [3]. The tetrahedral grid resolution is set to 128, and the batch size is set to 1.

Text-to-3D For the text-to-3D task, we use Stable Diffusion 2.1 base. The geometry is optimized for 3,000 iterations and the texture for another 1,000 iterations. Following Fantasia3D [3], during the initial training iterations, we concatenate the rendered normal and depth maps to serve directly as the latent code for the diffusion models, facilitating rapid convergence to a basic shape. As Magic3D [18] haven't released their code, we use the implementation from threestudio. To ensure a fair comparison with Fantasia3D [3], we also adopt the threestudio implementation.

Image-to-3D In the image-to-3D task, we use Stable Zero-1-to-3 and adapt our pipeline by incorporating Φ_t in the initial stage to encode the materials at the center of each tetrahedron. Specifically, we start by inferring the materials at the center of each tetrahedron. Next, we compute the per-tetrahedron PBR color $\mathbf{c}$ using the rendering equation for each tetrahedron. This per-tetrahedron PBR color $\mathbf{c}$ is subsequently passed to the rasterizer to perform the alpha-blending for the final output: $\mathcal{C} = \sum_{i \in N} T_i \alpha_i \mathbf{c}_i$. To further refine the output, we introduce an MSE loss that aligns the rendering color map $\mathcal{C}^r$ and opacity map $\mathcal{O}^r$ at reference view with the provided reference image $\tilde{\mathcal{C}}^r$ and mask $\tilde{\mathcal{O}}^r$: $\mathcal{L}_{ref} = \lambda_{rgb} \|\mathcal{C}^r - \tilde{\mathcal{C}}^r\|_2^2 + \lambda_{mask} \|\mathcal{O}^r - \tilde{\mathcal{O}}^r\|_2^2$. We set the loss weights λ_{rgb} and λ_{mask} to 10,000 and 1,000, respectively. Also, we decrease λ_{eik} and s_{ratio} to 100 and 2, respectively.

B.3 Evaluation with rich diffusion priors

This part is implemented based on the RichDreamer [34] codebase using the settings of DMTet [40]. The tetrahedral grid resolution is set to 256 and the batch size is set to 4 for two stages. We optimize the geometry for 3,000 steps, with the first 1,000 steps using latent code, followed by an additional 2,000 steps for texture optimization. While RichDreamer reports significantly increased stability at a rendering resolution of 1024, we achieve stable results at a lower resolution of 512. Therefore, we set our rendering resolution to 512.

CLIP score We adopt the evaluation process in RichDreamer [34] to calculate CLIP scores using the CLIP model [35] (vit-g-14). For geometry CLIP scores, we render generated meshes with uniform albedo and produce 16 different views for each object. The average CLIP scores are computed by discarding the highest and lowest scores from the provided text prompts. For texture CLIP scores, textured meshes are rendered. Since RichDreamer has not released the prompt list (113 objects) used for their metrics, we utilize an alternative list named “*prompts_dmtet.txt*” (183 objects) available on their official GitHub repository.

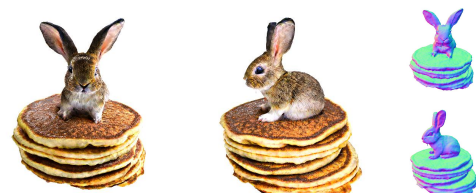
C More results

We present an extensive gallery of visual results in Figure 10-13.

D Discussions on the potential social impacts

The proposed *TeT-Splatting* method can make it easier for people to enter the animation and related industries. This can simplify production processes, reduce costs, and allow more people to create high-quality 3D assets. However, these improvements could also lead to job losses for professionals who work in traditional roles. To address this, it may be necessary to gradually adjust training programs and align the workforce with future demands.

Additionally, while our method improves the efficiency of 3D generation, it also carries the biases present in the foundational models we use. These models can have built-in biases related to race, gender, and culture, which might appear in the generated content, reinforcing stereotypes. The easier creation of realistic 3D models also raises concerns about copyright infringement and misuse, highlighting the need for strong ethical guidelines and regulatory frameworks.



"a zoomed out DSLR photo of a baby bunny sitting on top of a stack of pancakes"



"a DSLR photo of a bear dancing ballet"



"a DSLR photo of a bear dressed in medieval armor"



"a beautiful dress made out of garbage bags, on a mannequin. Studio lighting, high quality, high resolution"



"a zoomed out DSLR photo of a beautifully carved wooden knight chess piece"



"a zoomed out DSLR photo of a blue tulip"



"a brightly colored mushroom growing on a log"



"a DSLR photo of a cake covered in colorful frosting with a slice being taken out, high resolution"



"a capybara wearing a top hat, low poly"



"a DSLR photo of a cauldron full of gold coins"



"a completely destroyed car"



"a crocodile playing a drum set"

Figure 10: More results of *TeT-Splatting* with rich diffusion priors.

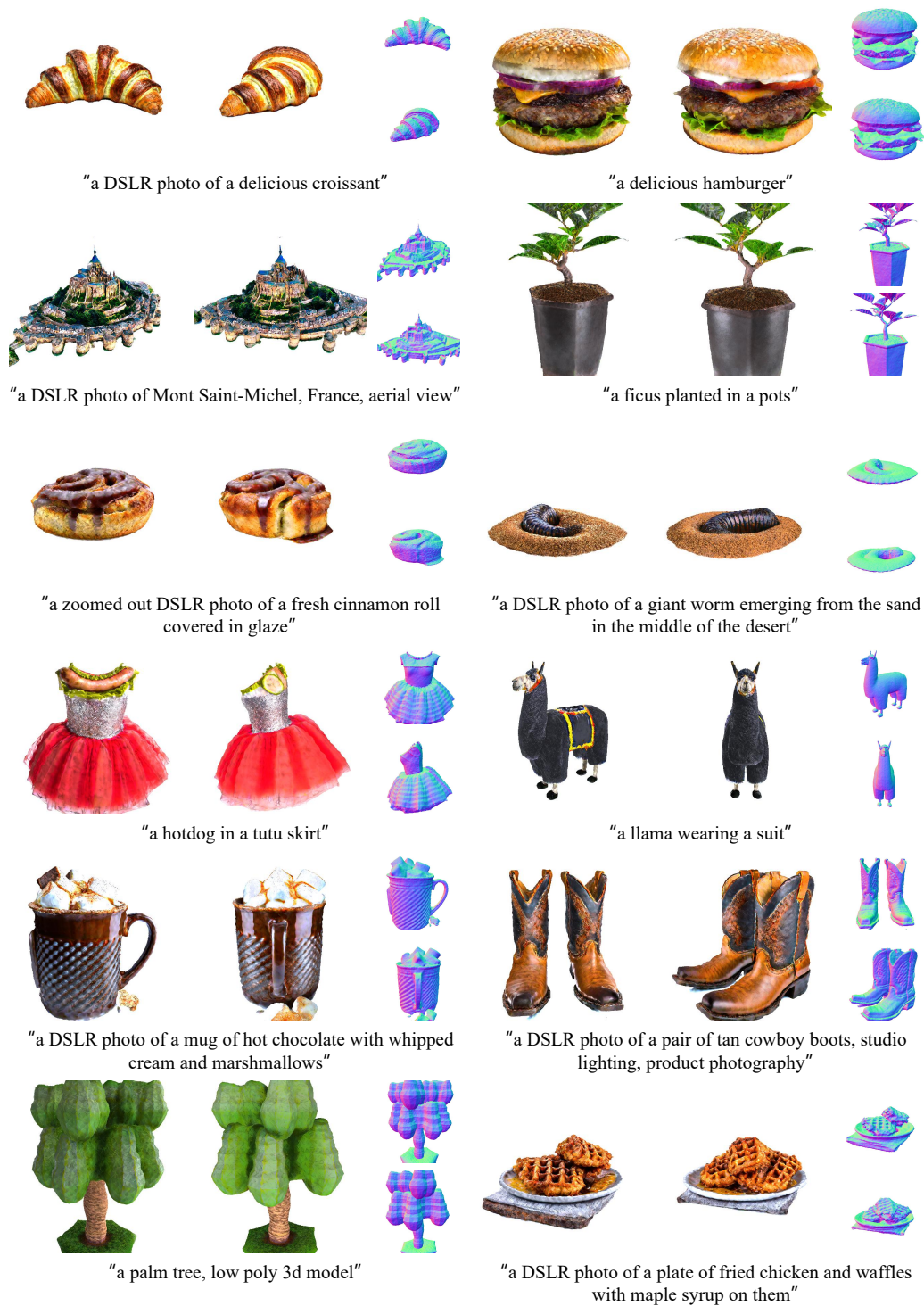


Figure 11: More results of *TeT-Splatting* with rich diffusion priors.



Figure 12: More results of *TeT-Splatting* with rich diffusion priors.



Figure 13: More results of *TeT-Splatting* with rich diffusion priors.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: The motivation and contributions of this study are clearly stated in the introduction, with a brief overview of them provided in the abstract.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: We discussed the limitations of this work in Appendix 6.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory Assumptions and Proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[Yes\]](#)

Justification: Our main theoretical result is the gradient of SDF inside a tetrahedron, which is proved in Appendix A.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental Result Reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We provided the detailed design and experimental setting in our paper.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Our results do not rely on any private data. We provide a link to our project page, which contains the code.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental Setting/Details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: The details are provided in Appendix A and B.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment Statistical Significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: We do not report statistically significant following the common practice in this field.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments Compute Resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: We report the generation time in Section 5 and the used GPU resources in Appendix B.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code Of Ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: We have reviewed the NeurIPS Code of Ethics and ensure that our research strictly conforms to the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader Impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The potential societal impacts are discussed in Appendix D.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We have cited the original paper of the used assets.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: The paper does not release new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.