Hash3D: Training-free Acceleration for 3D Generation

Xingyi Yang Songhua Liu Xinchao Wang*
National Unversity of Singapore
{xyang,songhua.liu}@u.nus.edu, xinchao@nus.edu.sg

Figure 1. Examples by applying our Hash3D on Gaussian-Dreamer [63] and Dream-Gaussian [53]. We accelerate Gaussian-Dreamer by $1.5\times$ and Dream-Gaussian by $4\times$ with comparable visual quality.

Abstract

The quality of 3D generative modeling has been notably improved by the adoption of 2D diffusion models. Despite this progress, the cumbersome optimization process per se *presents a critical problem to efficiency. In this paper, we introduce Hash3D, a universal acceleration for 3D score distillation sampling (SDS) without model training. Central to Hash3D is the observation that images rendered from similar camera positions and diffusion time-steps often have redundant feature maps. By hashing and reusing these feature maps across nearby timesteps and camera angles, Hash3D eliminates unnecessary calculations. We implement this through an adaptive grid-based hashing. As a result, it largely speeds up the process of 3D generation. Surprisingly, this feature-sharing mechanism not only makes generation faster but also improves the smoothness and view consistency of the synthesized 3D objects. Our experiments covering 5 text-to-3D and 3 image-to-3D models, demonstrate Hash3D's versatility to speed up optimization, enhancing efficiency by* $1.5 \sim 4\times$. *Additionally, Hash3D's integration with 3D Gaussian splatting largely speeds up 3D model creation, reducing text-to-3D conversion to about 10 minutes and image-to-3D conversion to 30 seconds. The project page is* `https://adamdad.github.io/hash3D/`.

1. Introduction

In the field of 3D generation, the integration of 2D diffusion models [36, 55] has led to notable advancements. These methods leverage off-the-the-shelf image diffusion models to distill 3D models by predicting 2D score functions at different views, known as score distillation sampling (SDS).

While this approach opens up new opportunities for creating realistic 3D assets, it also brings significant efficiency challenges. Particularly, SDS requires thousands of score predictions from different camera angles and denoising steps in the diffusion model. This results in long opti-

*Corresponding author.

mization times, sometimes taking hours to create a single object [57]. These durations hinder practical use. We need new solutions to improve its efficiency.

To mitigate this bottleneck, current efforts focus on three potential solutions. The first solution trains feed-forward models [7, 12, 18, 24, 61] to skip the lengthy optimization. While effective, this method requires extensive training time and substantial computational resources. The second approach [40, 53, 63] reduces optimization times by using faster 3D representations. However, each type of representation needs a unique design for 3D generation, which creates its own challenges. The third approach directly generates sparse views to model 3D objects [16, 27]. This method assumes near-perfect consistency for generated views, which, in practice, is often not achievable.

Focusing back on SDS, the main computational cost comes from repeatedly sampling the 2D image score function [51]. Inspired by techniques that accelerate 2D diffusion sampling [3, 28, 50], we posed the question: *Is it possible to reduce the number of inference steps of the diffusion model for 3D generation?*

In exploring this question, we make a crucial observation: denoising outputs and feature maps from near camera positions and timesteps are very similar. This discovery led us to develop **Hash3D**, which reduces the computation by leveraging this redundancy.

At its core, Hash3D stores and hashes previously computed features to reduce time. We do this using a a grid-based hash table. Specifically, when a new view is close to one that has already been processed, Hash3D retrieves and reuses the nearby features from the table. This reuse allows Hash3D to compute the current view's score function without repeating earlier calculations.Additionally, we developed a method to dynamically adjust the grid size for each view, which makes the system more adaptable. As a result, Hash3D saves computational resources without requiring any model training or complex changes, making it easy to implement and efficient to use.

Beyond improving efficiency, Hash3D also has the potential to improve the cross-view consistency of generated objects. Traditional diffusion-based methods often result in 3D objects with disjointed appearances when viewed from various angles [2]. In contrast, Hash3D links independently generated views by sharing features within each grid. It leads to smoother, more consistent 3D models.

Another key advantage of Hash3D is its versatility. It could be integrated into any diffusion-based 3D generative workflows. Our experiments, covering 5 text-to-3D and 3 image-to-3D models, demonstrate Hash3D's versatility to speed up optimization, enhancing efficiency by $1.5 \sim 4\times$, without compromising on performance. Specifically, the integration of Hash3D with 3D Gaussian Splatting [13] brings a significant leap forward, cutting down the time for text-to-3D to about 10 minutes and image-to-3D to roughly 30 seconds.

The contribution of this paper can be summarized into

- We introduce Hash3D, a versatile, plug-and-play and training-free acceleration method for diffusion-based text-to-3D and image-to-3D models.
- The paper emphasizes the redundancy in diffusion models when processing nearby views and timesteps. This finding motivates the development of Hash3D, aiming to boost efficiency without compromising quality.
- Hash3D employs an adaptive grid-based hashing to efficiently retrieve features, significantly reducing the computations across view and time.
- Our extensive testing demonstrates that Hash3D not only speeds up the generative process by $1.5 \sim 4\times$, but also results in a slight improvement in performance.

2. Related Work

3D Generation Model. Building 3D generative models is important but hard. Typically, these models are trained to predict the 3D representation parameters, such as voxel [59], point cloud [1, 34], implicit function [11], triplane [49, 61]. Despite these advances, scalability remains a major challenge due to the demands of large datasets and high computational costs. A promising solution is to leverage 2D generative models generate 3D models. Recent progress, especially in diffusion-based models using score distillation [36, 46, 47], have shown significant potential in 3D generation. However, these methods are heavily hindered by long optimization processes.

Efficient Diffusion Model. Diffusion models, while effective, are often time-consuming due to their iterative denoising process. Efforts to speed them up focus on two main strategies. The first reduces sampling steps using advanced sampling [3, 22, 28, 50] or distillation [45, 52] techniques. The second strategy aims to streamline each model inference by developing smaller models [9, 14, 62] or reusing features from nearby steps [20, 29, 30]. Although these methods are common in 2D and video generation, their application to 3D generative tasks remains largely unexplored.

Hashing Techniques. Hashing is a fast and efficient method for data storage. This is achieved by converting variable-sized inputs into fixed-size hash codes using *hash functions*. These codes index a *hash table* for fast and consistent data access. Commonly used in file systems, hashing is also effective in 3D representation [10, 33, 35, 60], neural network compression [6, 15], and as components in deep networks [41] or for developing neural hash functions [4, 17, 19, 65]. Our study explores the application of hashing to retrieve features from 3D generation. By adopting this technique, we aim to reduce computational overhead for repeated diffusion sampling and speed up the creation of realistic 3D objects.

3. Preliminary

In this section, we provide the notations and background on optimization-based 3D generation, focusing on diffusion models and Score Distillation Sampling (SDS) [36].

3.1. Diffusion Models

Diffusion models are generative models that reverse a noise-adding process through a series of latent variables. Starting with data $\mathbf{x}_0 \sim q(\mathbf{x}_0)$, Gaussian noise is progressively added over T steps during the forward process, each defined by $q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$, where $\beta_t \in [0,1]$. Due to the Gaussian nature, $\mathbf{x}_t$ can be directly sampled as:

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I}) \tag{1}$$

where $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$

The reverse process is modeled as a Markov chain parameterized by a denoising neural network $\boldsymbol{\epsilon}(\mathbf{x}_t, t, y)$, where y is the conditional input, such as text [44] or camera pose [26]. The training of the denoiser aims to minimize a re-weighted evidence lower bound (ELBO), aligning with the noise:

$$\mathcal{L}_{\text{DDPM}} = \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[||\boldsymbol{\epsilon} - \boldsymbol{\epsilon}(\mathbf{x}_t, t, y)||_2^2\right] \tag{2}$$

Here, $\boldsymbol{\epsilon}(\mathbf{x}_t, t, y)$ approximates the score function $\nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t|\mathbf{x}_0)$. Data generation is achieved by denoising from noise, often enhanced using classifier-free guidance with scale parameter ω: $\hat{\epsilon}(\mathbf{x}_t, t, y) = (1+\omega)\boldsymbol{\epsilon}(\mathbf{x}_t, t, y) - \omega\boldsymbol{\epsilon}(\mathbf{x}_t, t, \emptyset)$.

Extracting Feature from Diffusion Model. A diffusion denoiser $\boldsymbol{\epsilon}$ is typically parameterized with a U-Net [43]. It uses l down-sampling layers $\{D_i\}_{i=1}^{l}$ and up-sampling layers $\{U_i\}_{i=1}^{l}$, coupled with skip connections that link features from D_i to U_i. This module effectively merges high-level features from U_{i+1} with low-level features from D_i, as expressed by the equation:

$$\mathbf{v}_{i+1}^{(U)} = \text{concat}(D_i(\mathbf{v}_{i-1}^{(D)}), U_{i+1}(\mathbf{v}_i^{(U)})) \tag{3}$$

In this context, $\mathbf{v}_i^{(U)}$ and $\mathbf{v}_{i+1}^{(D)}$ represent the up-sampled and down-sampled features after the i-th layer, respectively.

3.2. Score Distillation Sampling (SDS)

The Score Distillation Sampling (SDS) [36] represents an optimization-based 3D generation method. This method focuses on optimizing the 3D representation, denoted as Θ, using a pre-trained 2D diffusion models with its noise prediction network, denoted as $\boldsymbol{\epsilon}_{\text{pretrain}}(x_t, t, y)$.

Given a camera pose $\boldsymbol{c} = (\theta, \phi, \rho) \in \mathbb{R}^3$ defined by elevation ϕ, azimuth θ and camera distances ρ, and the its corresponding prompt y^c, a differentiable rendering function $g(\cdot;\Theta)$, SDS aims to refine the parameter Θ, such that each rendered image $\boldsymbol{x}_0 = g(\boldsymbol{c};\theta)$ is perceived as realistic by $\boldsymbol{\epsilon}_{\text{pretrain}}$. The optimization objective is formulated as follows:

$$\min_{\Theta} \mathcal{L}_{\text{SDS}} = \mathbb{E}_{t,\boldsymbol{c}}\left[\frac{\sigma_t}{\alpha_t}\omega(t)\text{KL}\left(q^{\Theta}(\boldsymbol{x}_t|y_c, t) \,\|\, p(\boldsymbol{x}_t|y_c; t)\right)\right] \tag{4}$$

By excluding the Jacobian term of the U-Net, the gradient of the optimization problem can be effectively approximated:

$$\nabla_{\Theta}\mathcal{L}_{\text{SDS}} \approx \mathbb{E}_{t,\boldsymbol{c},\boldsymbol{\epsilon}}\left[\omega(t)(\boldsymbol{\epsilon}_{\text{pretrain}}(\boldsymbol{x}_t, t, y^c) - \boldsymbol{\epsilon})\frac{\partial \boldsymbol{x}}{\partial \Theta}\right] \tag{5}$$

To optimize Eq. 5, we randomly sample different time-step t, camera $\boldsymbol{c}$, and random noise $\boldsymbol{\epsilon}$, and compute gradient of the 3D representation, and update θ accordingly. This approach ensures that the rendered image from 3D object aligns with the distribution learned by the diffusion model.

Efficiency Problem. The main challenge lies in the need for thousands to tens of thousands of iterations to optimize Eq 5, each requiring a separate diffusion model inference. This process is time-consuming due to the model's complexity. We make it faster by using a hash function to reuse features from similar inputs, cutting down on the number of calculations needed.

4. Hash3D

This section introduces Hash3D, a plug-and-play tool that enhances the efficiency of SDS. We start by analyzing the redundancy presented in the diffusion model when performing 3D generation. Based on the finding, we present our strategy that employs a grid-based hashing to reuse feature across different sampling iterations.

4.1. Probing the Redundancy in SDS

Typically, SDS randomly samples camera poses and timesteps to ensure that the rendered views align with the diffusion model's distribution. However, during this repeated sampling, we observe that deep feature extraction at proximate $\boldsymbol{c}$ and t often reveals a high degree of similarity. Therefore, this similarity underpins our method, suggesting that reusing features from nearby points does not significantly impact the model's predictions.

Measuring Similarity. Intuitively, images captured from similar camera positions and at similar times result in similar visual content. We hypothesize that features produced by diffusion models exhibit a similar pattern. Specifically, we propose two hypotheses: (1) *temporal similarity*: features extracted at close timesteps are similar, and (2) *spatial similarity*: features extracted from images rendered at close camera poses are similar.

Regarding the *temporal similarity*, previous studies [20, 30] have noted that features extracted from adjacent

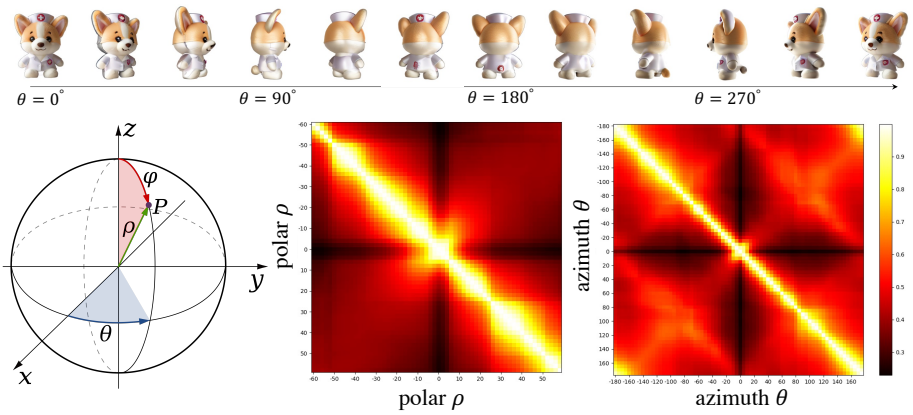

Figure 2. Feature similarity extracted from different camera poses.

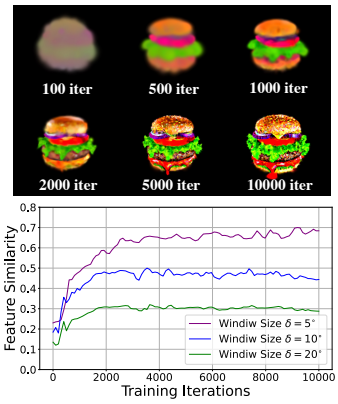

Figure 3. Similarity dynamics at different SDS training steps.

timesteps in diffusion models show a high level of similarity. To test the hypothesis of *spatial similarity*, we conducted two preliminary studies measuring feature similarity when denoising images of the same object taken from different camera poses.

Feature Similarity in Camera-Conditioned Diffusion. In the first study, we used Zero-123 [26], a model that generates images from different camera poses conditioned on a single input image. For each specific camera angle and timestep, we extracted features $\mathbf{v}_{l-1}^{(U)}$ from the input of the last up-sampling layer. By varying the elevation (ϕ) and azimuth (θ) angles, we measured the cosine similarity between features of different views, averaging the results across all timesteps.

As shown in Figure 2, features from views within a $[-10°, 10°]$ range for both elevation and azimuth have high similarity scores, with most values exceeding 0.8.

Feature Similarity Dynamics in T2I diffusion. In the second study, we test the feature redundancy using text-to-image diffusion models. During the SDS process, the optimized 3D object gets continuously updated. This causes the similarity of rendered images to vary across different training steps. We analyze this similarity by denoising the rendered images at various stages using a diffusion model.

Specifically, at different SDS iterations, we render images uniformly at camera angles $(\theta, \phi) = (10°, 0°, 5°, \ldots, 360°)$. We add a noise level $\sqrt{\bar{\alpha}_t} = 0.5$, process them with Stable Diffusion [42], and extract features $\mathbf{v}_{l-1}^{(U)}$. We then compute the average cosine similarity within windows of $\delta = \{5°, 10°, 20°\}$ for all feature pairs. The SDS process runs for 10,000 iterations.

As illustrated in Figure 3, the object appears blurry with low feature similarity across views in the early stages of training. By approximately 1,000 iterations, the images become clearer, and feature similarity sharply increases. For example, when $\delta = 5°$, the similarity fluctuates around 0.65. However, with larger windows, the similarity remains lower compared to smaller windows. These results highlight the redundancy in predicted outputs throughout the SDS process.

Based on these analysis, we present our approach: instead of computing the noise prediction for every new camera pose and timestep, we create a memory system to store previously computed features. As such, we can retrieve and reuse these pre-computed features whenever needed. Ideally, this approach could reduces redundant calculations and speeds up the optimization process.

4.2. Hashing-based Feature Reuse

Based on our analysis, we developed Hash3D, which uses hashing to speedup SDS. Hash3D reduces the repetitive computational cost in diffusion models by trading storage space for faster 3D optimization.

At its core, Hash3D employs a hash table to store and retrieve previously computed features. When Hash3D samples a specific camera pose $\boldsymbol{c}$ and timestep t, it first checks the hash table for similar features. If a match is found, it's reused directly in the diffusion model, significantly cutting down on computation. If not, it performs standard inference and adds the new features to the hash table for future use.

Grid-based Hashing. To efficiently index the hash table, we use *grid-based hashing* based on camera poses $\boldsymbol{c} = (\theta, \phi, \rho)$ and timestep t. This function assigns each $\boldsymbol{c}$ and t to a grid cell for data storage and retrieval.

Firstly, we define the size of our grid cells in both the spatial and temporal domains, denoted as $\Delta\theta, \Delta\phi, \Delta\rho$ and Δt respectively. For each input key $[\theta, \phi, \rho, t]$, we calculate the grid cell indices:

$$i = \left\lfloor \frac{\theta}{\Delta\theta} \right\rfloor, j = \left\lfloor \frac{\phi}{\Delta\phi} \right\rfloor, k = \left\lfloor \frac{\rho}{\Delta\rho} \right\rfloor, l = \left\lfloor \frac{t}{\Delta t} \right\rfloor \quad (6)$$

These indices are combined into a single hash code: $\texttt{idx} = (i + N_1 \cdot j + N_2 \cdot k + N_3 \cdot l) \mod n$ is used, where N_1, N_2, N_3

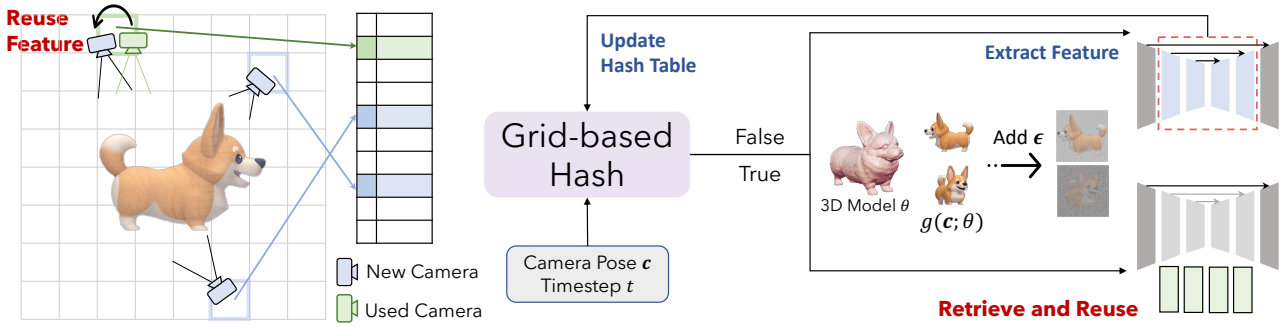

Figure 4. Overall pipeline of our Hash3D. Given the sampled camera and time-step, we retrieve the intermediate diffusion feature from hash table. If no matching found, it performs a standard inference and stores the new feature in the hash table; otherwise, if a feature from a close-up view already exists, it is reused without re-calculation.

are large prime numbers [35, 54], and n denotes the size of the hash table. This hash function maps keys with similar camera poses and timesteps to the same bucket. This grid-based approach not only speeds up data retrieval but also preserves the spatial-temporal relationships in the data, which is crucial for our method.

Collision Resolution. When multiple keys are assigned to the same hash value, a collision occurs. We address these collisions using *separate chaining*. In this context, each hash value $\texttt{idx}$ is linked to a distinct queue, denoted as $q_{\texttt{idx}}$. To ensure the queue reflects the most recent data and remains manageable in size, it is limited to a maximum length $Q = 3$. When this limit is reached, the oldest elements is removed to accommodate the new entry, ensuring the queue stays relevant to the evolving 3D representation.

Feature Retrieval and Update. After computing the hash value $\texttt{idx}$, we either retrieve features from the hash table or update it with new ones. We control this with hash probability $0 < \eta < 1$. With probability η, we retrieve features; otherwise, we perform an update.

For feature updates, following prior work [30], we extract the feature $\mathbf{v}_{l-1}^{(U)}$, which is the input of the last upsampling layer in the U-net. Once extracted, we compute the hash code $\texttt{idx}$ and append the data to the corresponding queue $q_{\texttt{idx}}$. The stored data includes noisy latent input $\boldsymbol{x}$, camera pose $\boldsymbol{c}$, timestep t, and extracted diffusion features $\mathbf{v}_{l-1}^{(U)}$.

For feature retrieval, we aggregate data from $q_{\texttt{idx}}$ through weighted averaging. This method considers the distance of each noisy input $\boldsymbol{x}_i$ from the current query point $\boldsymbol{x}$. The weighted average $\mathbf{v}$ for a given index is calculated as follows:

$$\mathbf{v} = \sum_{i=1}^{|q_{\texttt{idx}}|} W_i \mathbf{v}_i, \text{ where } W_i = \frac{e^{(-||\boldsymbol{x}-\boldsymbol{x}_i||_2^2)}}{\sum_{i=1}^{|q_{\texttt{idx}}|} e^{(-||\boldsymbol{x}-\boldsymbol{x}_i||_2^2)}} \tag{7}$$

Here, W_i is the weight assigned to $\mathbf{v}_i$ based on its distance from the query point, and $|q_{\texttt{idx}}|$ is the current length of the queue. An empty queue $|q_{\texttt{idx}}|$ indicates unsuccessful retrieval, necessitating feature update.

Comparison with Feature Caching. Our hashing-based feature reuse is fundamentally different from previous works that use feature caching [20, 30, 58] to accelerate diffusion models. Feature caching assumes an *ordered* denoising process, while SDS involves an inherently **unordered** and stochastic sampling process. Moreover, storing all features is costly. To handle this, we employ a hash table to organize these features efficiently. As a result, those feature caching methods *cannot be directly applied* to our setup.

4.3. Adaptive Grid Hashing

In grid-based hashing, the selection of an appropriate grid size $\Delta\theta, \Delta\phi, \Delta\rho, \Delta t$ — plays a pivotal role. As illustrated in Section 4.1, we see three insights related to grid size. First, feature similarity is only maintained at a median grid size; overly large grids tend to produce artifacts in generated views. Second, it is suggested that ideal grid size differs across various objects. Third, even for a single object, optimal grid sizes vary for different views and time steps, indicating the necessity for adaptive grid sizing to ensure optimal hashing performance.

Learning to Adjust the Grid Size. To address these challenges, we propose to dynamically adjusting grid sizes. The objective is to maximize the average cosine similarity $\cos(\cdot,\cdot)$ among features within each grid. In other words, only if the feature is similar enough, we can reuse it. Such problem is formulated as

$$\max_{\Delta\theta,\Delta\phi,\Delta\rho,\Delta t} \frac{1}{|q_{\texttt{idx}}|} \sum_{i,j}^{|q_{\texttt{idx}}|} \cos(\mathbf{v}_j, \mathbf{v}_i), \quad s.t. |q_{\texttt{idx}}| > 0 \quad \text{[Non-empty]} \tag{8}$$

Given that our hashing function is *non-differentiable*, we employ a brute-force approach. Namely, we evaluate M predetermined potential grid sizes, each corresponding to a distinct hash table, and only use best one.

For each input $[\theta, \phi, \rho, t]$, we calculate the hash code $\{\texttt{idx}^{(m)}\}_{m=1}^{M}$ for M times, and indexing in each bucket. Feature vectors are updated accordingly, with new elements being appended to their respective bucket. We calculate the

Figure 5. Qualitative Results using Hash3D along with Zero123 for image-to-3D generation. We mark the visual dissimilarity in yellow.

cosine similarity between the new and existing elements in the bucket, maintaining a running average $s_{\texttt{idx}^{(n)}}$ of these similarities

$$s_{\texttt{idx}^{(m)}} \leftarrow \gamma s_{\texttt{idx}^{(m)}} + (1-\gamma)\frac{1}{|q_{\texttt{idx}^{(m)}}|}\sum_{i=1}^{|q_{\texttt{idx}^{(m)}}|}\cos(\mathbf{v}_{new}, \mathbf{v}_i) \tag{9}$$

During retrieval, we hash across all M grid sizes but only consider the grid with the highest average similarity for feature extraction.

Computational and Memory Efficiency. Despite employing a brute-force approach that involves hashing M times for each input, our method maintains computational efficiency due to the low cost of hashing. It also maintains memory efficiency, as hash tables store only references to data. To prioritize speed, we deliberately avoid using neural networks for hashing function learning.

5. Experiment

In this section, we evaluate Hash3D by integrating it with various 3D generative models, encompassing both image-to-3D and text-to-3D tasks.

5.1. Experimental Setup

Baselines. To verify our method, we conduct extensive tests across a wide range of baseline text-to-3D and image-to-3D methods.

- **Image-to-3D.** We build our method on Zero-123+SDS [25], DreamGaussian [53] and Magic123 [38]. For Zero-123+SDS, we incorporate Instant-NGP [32] and Gaussian Splatting [13] as its representation. We call these two variants Zero-123 (NeRF) and Zero-123 (GS).
- **Text-to-3D.** Our tests also covered a range of methods, such as Dreamfusion [36], Fantasia3D [5], Latent-NeRF [31], Magic3D [21], and GaussianDreamer [63].

For DreamGaussian and GaussianDreamer, we implement Hash3D on top of the official code. And for other methods, we use the reproduction from `threestudio`*.

Implementation Details. We stick to the same hyper-parameter setup within their original implementations of these methods. For text-to-3D, we use the `stable-diffusion-2-1`† as our 2D diffusion model. For image-to-3D, we employ the `stable-zero123`‡. We use a default hash probability setting of $\eta = 0.1$. We use $M = 3$ sets of grid sizes, with $\Delta\theta, \Delta\phi, \Delta t \in \{10, 20, 30\}$ and $\Delta\rho \in \{0.1, 0.15, 0.2\}$. We verify this hyper-parameter setup in the ablation study.

Dataset and Evaluation Metrics. To assess our method, we focus on evaluating the computational cost and visual quality achieved by implementing Hash3D.

- **Image-to-3D.** For image-to-3D experiments, we used the Google Scanned Objects (GSO) dataset [8] for evaluation [24, 26]. We evaluated novel view synthesis (NVS) performance with PSNR, SSIM [56], and LPIPS [64]. We selected 30 objects, each with a 256^2 input image for 3D reconstruction. We rendered 16 views at a 30-degree elevation with varying azimuths to compare the reconstructions with ground truth. CLIP-similarity scores were calculated to ensure semantic consistency between rendered views and original images.
- **Text-to-3D.** We generated 3D models from 50 different prompts selected from DreamFusion. To evaluate our methods, we focus on two primary metrics: mean±std CLIP-similarity [23, 37, 39] and the average generation time for each method. CLIP-similarity was measured between the input prompt and 8 uniformly rendered views.
- **User Study.** To evaluate the visual quality of generated 3D objects, we conducted a study with 44 participants. They

*https://github.com/threestudio-project/threestudio
†https://huggingface.co/stabilityai/stable-diffusion-2-1
‡https://huggingface.co/stabilityai/stable-zero123

Table 1. Speed and quality comparison when applying Hash3D on image-to-3D task. We report the time from original papers.

Method	Time↓	Speed↑	MACs↓	PSNR↑	SSIM↑	LPIPS↓	CLIP-G/14↑
DreamGaussian	2m	-	169G	16.202	0.772	0.225	0.693
+ **Hash3D**	**30s**	**4.0×**	**154G**	**16.356**	**0.776**	**0.223**	**0.694**
Zero-123(NeRF)	20m	-	169G	17.773	0.787	0.198	0.662
+ **Hash3D**	**7m**	**3.3×**	**154G**	**17.961**	**0.789**	**0.196**	**0.665**
Zero-123(GS)	6m	-	169G	18.409	0.789	**0.204**	**0.643**
+ **Hash3D**	**3m**	**2.0×**	**154G**	**18.616**	**0.793**	**0.204**	0.632
Magic123	120m	-	847G	**18.718**	**0.803**	**0.169**	**0.718**
+ **Hash3D**	**74m**	**1.6×**	**777G**	18.631	**0.803**	0.174	0.715

viewed 12 video renderings from two methods: Zero-123 (NeRF) for images-to-3D and Gaussian-Dreamer for text-to-3D, with and without Hash3D. Participants rated each pair by distributing 100 points to indicate perceived quality differences.

- **Computational Cost.** We report the running time for each experiment on a single RTX A5000 and include MACs in the tables. As feature retrieval is stochastic, we provide the theoretical average MACs, assuming all retrievals succeed.

5.2. 3D Generation Results

Image-to-3D Qualitative Results. Figure 5 shows the results of integrating Hash3D into the Zero-123 framework for generating 3D objects. This integration maintains visual quality and view consistency while significantly reducing processing time. In some cases, Hash3D outperforms the baseline, such as the clearer "dragon wing boundaries" in row 1 and the more distinct "train taillights" in row 3. Similar visual fidelity is seen in Figure 1, where Hash3D is used with DreamGaussian, demonstrating effective quality maintenance and improved efficiency.

Image-to-3D Quantitative Results. Table 1 presents a detailed numerical analysis of novel view synthesis, including CLIP scores and running times for all four baseline methods. Notably, Our method achieves a $4\times$ speedup on DreamGaussian and $3\times$ on Zero-123 (NeRF), due to Hash3D's efficient feature retrieval and reuse. This not only accelerates processing but also slightly improves CLIP score performance by sharing features across views, reducing inconsistencies, and producing smoother 3D models.

Text-to-3D Qualitative Results. In Figure 6, we present the results generated by our Hash3D, on top of DreamFusion [36], SDS+GS, and Fantasia3D [5]. It demonstrates that Hash3D maintains comparable visual quality to these established methods.

Text-to-3D Quantitative Results. Table 2 presents a quantitative evaluation of Hash3D. Hash3D significantly reduces processing times across various methods while maintaining

Table 2. Speed and quality comparison between various text-to-3D baselines integrated with Hash3D.

Method	Time↓	Speed↑	MACs↓	CLIP-G/14↑	CLIP-L/14↑	CLIP-B/32↑
Dreamfusion	60m	-	679G	0.407	**0.267**	**0.314**
+ **Hash3D**	**40m**	**1.5×**	**622G**	**0.411**	0.266	0.312
Latent-NeRF	30m	-	679G	**0.406**	0.254	0.306
+ **Hash3D**	**17m**	**1.8×**	**622G**	**0.406**	**0.258**	**0.305**
SDS+GS	78m	-	679G	**0.413**	**0.263**	**0.313**
+ **Hash3D**	**40m**	**1.9×**	**622G**	0.402	0.252	0.306
Magic3D	90m	-	679G	**0.399**	**0.257**	**0.303**
+ **Hash3D**	**60m**	**1.5×**	**622G**	0.393	0.250	0.304
GaussianDreamer	15m	-	679G	0.412	0.267	0.312
+ **Hash3D**	**10m**	**1.5×**	**622G**	**0.416**	**0.271**	**0.312**

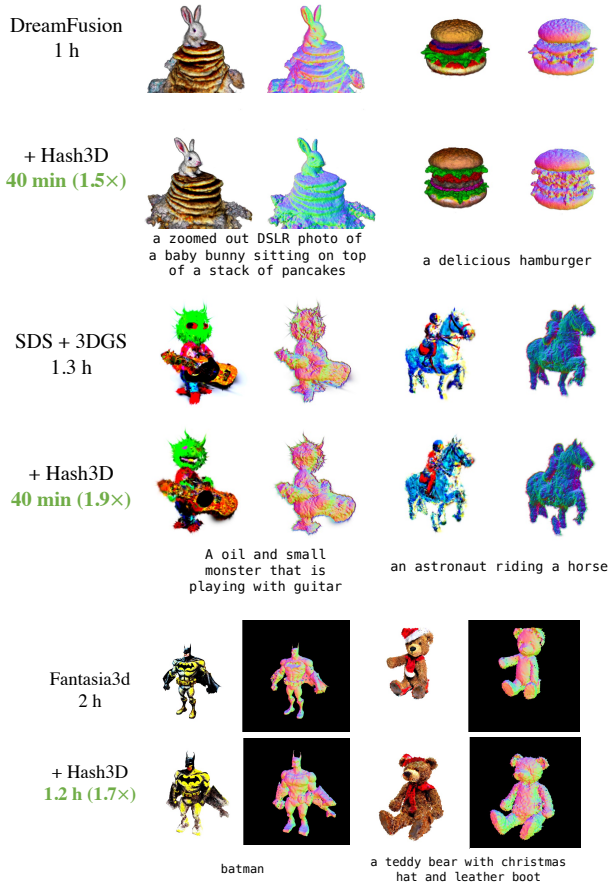

Figure 6. Visual comparison for text-to-3D task, when applying Hash3D to DreamFusion [36], SDS+GS and Fantasia3D [5].

visual quality, with minimal impact on CLIP scores. For methods like GaussianDreamer, it even slightly improves visual fidelity, indicating the benefit of leveraging relationships between nearby camera views.

User preference study. As shown in Figure 9, Hash3D received an average preference score of 52.33/100 and 56.29/100 when compared to Zero-123 (NeRF) and Gaussian-Dreamer. These scores are consistent with previous results, indicating that Hash3D slightly enhances the

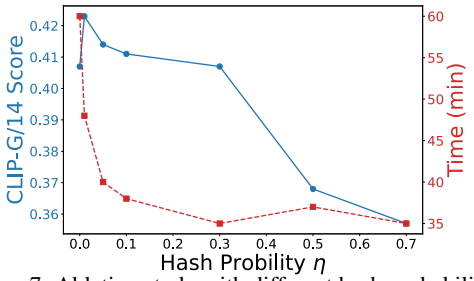

Figure 7. Ablation study with different hash probability η.

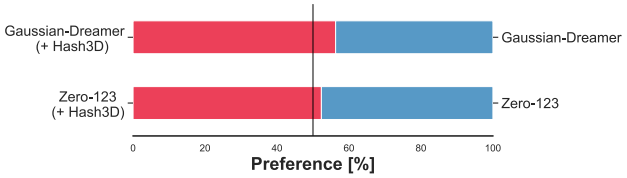

Figure 9. User preference study for Hash3D.

visual quality of the generated objects.

5.3. Ablation Study and Analysis

In this section, we study several key components in our Hash3D framework.

Ablation 1: Hashing *vs.* Storing All Features. We compare hashing features with storing all past features and retrieving them by similarity. As shown in Table 3, hashing is more effective. On efficiency side, storing all feature even causes an OOM error in Dreamfusion. Hashing requires only constant space. Additionally, our grid-based hashing leverages geometric information to improve sample quality. More visual results are available in the appendix.

Ablation 2: Hashing Features *vs.* Hashing Noise. In Hash3D, we hash intermediate features within the diffusion U-Net. Alternatively, we developed Hash3D with noise (Hash3D w/n), which hashes and reuses the denoising prediction directly. We tested both methods on the image-to-3D task using Zero123, with results shown in Table 8. Interestingly, while Hash3D w/n reduced processing time, it significantly lowered CLIP scores. This highlights that hashing features is more effective than hashing noise predictions.

Ablation 3: Influence of Hash Probability η. A key parameter in Hash3D is the retrieval probability η. We tested $\eta \in \{0.01, 0.05, 0.1, 0.3, 0.5, 0.7\}$ using Dreamfusion. As shown in Figure 7, runtime decreases as η increases. For $\eta < 0.3$, Hash3D slightly improved the visual quality of 3D models by enabling smoother noise predictions through feature sharing. However, for $\eta > 0.3$, the runtime gains were minimal. This balance of performance and efficiency led us to choose $\eta = 0.1$ for our main experiments.

Ablation 4: Adaptive Grid Size. We use AdaptGrid to dynamically adjusts the grid size for hashing based on each sample. Compared to using a constant grid size in Dreamfusion, AdaptGrid performs better as shown in Table 4. Larger grid sizes reduce the visual quality of 3D objects, while smaller grid sizes maintain quality but increase computation time because fewer features match. AdaptGrid effectively balances visual quality and efficiency by optimizing the grid size for each sample.

Method	**Time**	**CLIP-G/14**
Zero-123 (NeRF) + Hash3D w/n	6 min	0.631
Zero-123 (NeRF) + Hash3D	7 min	**0.665**
Zero-123 (GS) + Hash3D w/n	3 min	0.622
Zero-123 (GS) + Hash3D	3 min	**0.632**

Figure 8. Comparison between Hashing Features *vs.* Hashing Noise, applied to Zero-123.

Table 3. Comparison of feature retrieval with and without hashing.

Name	Time↓	GPU Mem.↓	CLIP-G/14↑
Hash3D+Zero-123 (NeRF) w/o hashing	11m	8G	0.661
Hash3D+Zero-123 (NeRF)	**7m**	**6G**	**0.665**
Hash3D+DreamFusion w/o hashing	-	OOM	-
Hash3D+DreamFusion	**40m**	**8G**	**0.411**

Table 4. Ablation study on the Adaptive *v.s.* Constant Grid Size.

$\Delta\theta, \Delta\phi, \Delta\rho, \Delta t$	(10, 10, 0.1, 10)	(20, 20, 0.15, 20)	(30, 30, 0.2, 30)	AdaptGrid (Ours)
CLIP-G/14↑	0.408	0.345	0.287	**0.411**
Time↓	48m	38m	32m	40m

Ablation 5: Spatial vs. Temporal Grid. A key distinction from previous feature caching acceleration methods is our incorporation of spatial redundancy in 3D space. To isolate the effects of spatial and temporal redundancy, we conducted an experiment on DreamFusion where we applied either the spatial grid or the temporal grid individually. As shown in Table 5, using only the temporal grid—where features are shared only across nearby timesteps—yields significant performance degradation.

Table 5. Ablation study on Spatial Grid *v.s.* Temporal Grid.

Spatial Grid ($\Delta\theta, \Delta\phi, \Delta\rho$)	Temporal Grid (Δt)	**CLIP-G/14↑**	**Time↓**
✓	✗	0.347	36m
✗	✓	0.280	30m
✓	✓	**0.411**	40m

6. Conclusion

In this paper, we present Hash3D, a training-free technique that improves the efficiency of diffusion-based 3D generative modeling. Hash3D utilizes adaptive grid-based hashing to efficiently retrieve and reuse features from adjacent camera poses, to minimize redundant computations. As a result, Hash3D not only speeds up 3D model generation by $1.5 \sim 4\times$ without the need for additional training, but it also improves the smoothness and consistency of the generated 3D models.

Acknowledgement

This project is supported by the Singapore Ministry of Education Academic Research Fund Tier 1 (WBS: A-0009440-01-00), and the National Research Foundation, Singapore, under its Medium Sized Center for Advanced Robotics Technology Innovation.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *International conference on machine learning*, pages 40–49. PMLR, 2018.

[2] Mohammadreza Armandpour, Huangjie Zheng, Ali Sadeghian, Amir Sadeghian, and Mingyuan Zhou. Re-imagine the negative prompt algorithm: Transform 2d diffusion into 3d, alleviate janus problem and beyond. *arXiv preprint arXiv:2304.04968*, 2023.

[3] Fan Bao, Chongxuan Li, Jun Zhu, and Bo Zhang. Analytic-DPM: an analytic estimate of the optimal reverse variance in diffusion probabilistic models. In *International Conference on Learning Representations*, 2022.

[4] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Philip S Yu. Hashnet: Deep learning to hash by continuation. In *Proceedings of the IEEE international conference on computer vision*, pages 5608–5617, 2017.

[5] Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[6] Wenlin Chen, James T. Wilson, Stephen Tyree, Kilian Q. Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, page 2285–2294. JMLR.org, 2015.

[7] Yiming Chen, Zhiqi Li, and Peidong Liu. Et3d: Efficient text-to-3d generation via multi-view distillation, 2023.

[8] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B McHugh, and Vincent Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2553–2560. IEEE, 2022.

[9] Gongfan Fang, Xinyin Ma, and Xinchao Wang. Structural pruning for diffusion models. In *Advances in Neural Information Processing Systems*, 2023.

[10] Sharath Girish, Abhinav Shrivastava, and Kamal Gupta. Shacira: Scalable hash-grid compression for implicit neural representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17513–17524, 2023.

[11] Heewoo Jun and Alex Nichol. Shap-e: Generating conditional 3d implicit functions. *arXiv preprint arXiv:2305.02463*, 2023.

[12] Heewoo Jun and Alex Nichol. Shap-e: Generating conditional 3d implicit functions, 2023.

[13] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023.

[14] Bo-Kyeong Kim, Hyoung-Kyu Song, Thibault Castells, and Shinkook Choi. Bk-sdm: A lightweight, fast, and cheap version of stable diffusion. *arXiv preprint arXiv:2305.15798*, 2023.

[15] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020.

[16] Xin Kong, Shikun Liu, Xiaoyang Lyu, Marwan Taher, Xiaojuan Qi, and Andrew J Davison. Eschernet: A generative model for scalable view synthesis. *arXiv preprint arXiv:2402.03908*, 2024.

[17] H. Lai, Y. Pan, Ye Liu, and S. Yan. Simultaneous feature learning and hash coding with deep neural networks. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3270–3278, Los Alamitos, CA, USA, 2015. IEEE Computer Society.

[18] Jiahao Li, Hao Tan, Kai Zhang, Zexiang Xu, Fujun Luan, Yinghao Xu, Yicong Hong, Kalyan Sunkavalli, Greg Shakhnarovich, and Sai Bi. Instant3d: Fast text-to-3d with sparse-view generation and large reconstruction model. *https://arxiv.org/abs/2311.06214*, 2023.

[19] Qi Li, Zhenan Sun, Ran He, and Tieniu Tan. Deep supervised discrete hashing. *Advances in neural information processing systems*, 30, 2017.

[20] Senmao Li, Joost van de Weijer, Fahad Khan, Tao Liu, Linxuan Li, Shiqi Yang, Yaxing Wang, Ming-Ming Cheng, et al. Faster diffusion: Rethinking the role of the encoder for diffusion model inference. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

[21] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[22] Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. In *International Conference on Learning Representations*, 2022.

[23] Minghua Liu, Chao Xu, Haian Jin, Linghao Chen, Zexiang Xu, Hao Su, et al. One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization. *arXiv preprint arXiv:2306.16928*, 2023.

[24] 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. *Advances in Neural Information Processing Systems*, 36, 2024.

[25] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9298–9309, 2023.

[26] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9298–9309, 2023.

[27] Yuan Liu, Cheng Lin, Zijiao Zeng, Xiaoxiao Long, Lingjie Liu, Taku Komura, and Wenping Wang. Syncdreamer: Generating multiview-consistent images from a single-view image. In *The Twelfth International Conference on Learning Representations*, 2024.

[28] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022.

[29] Xinyin Ma, Gongfan Fang, Michael Bi Mi, and Xinchao Wang. Learning-to-cache: Accelerating diffusion transformer via layer caching. *Advances in Neural Information Processing Systems*, 37:133282–133304, 2024.

[30] Xinyin Ma, Gongfan Fang, and Xinchao Wang. Deepcache: Accelerating diffusion models for free. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[31] G. Metzer, E. Richardson, O. Patashnik, R. Giryes, and D. Cohen-Or. Latent-nerf for shape-guided generation of 3d shapes and textures. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12663–12673, Los Alamitos, CA, USA, 2023. IEEE Computer Society.

[32] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, 2022.

[33] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022.

[34] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generating 3d point clouds from complex prompts. *arXiv preprint arXiv:2212.08751*, 2022.

[35] Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3d reconstruction at scale using voxel hashing. *ACM Trans. Graph.*, 32(6), 2013.

[36] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *The Eleventh International Conference on Learning Representations*, 2023.

[37] Guocheng Qian, Jinjie Mai, Abdullah Hamdi, Jian Ren, Aliaksandr Siarohin, Bing Li, Hsin-Ying Lee, Ivan Skorokhodov, Peter Wonka, Sergey Tulyakov, and Bernard Ghanem. Magic123: One image to high-quality 3d object generation using both 2d and 3d diffusion priors. *arXiv preprint arXiv:2306.17843*, 2023.

[38] Guocheng Qian, Jinjie Mai, Abdullah Hamdi, Jian Ren, Aliaksandr Siarohin, Bing Li, Hsin-Ying Lee, Ivan Skorokhodov, Peter Wonka, Sergey Tulyakov, and Bernard Ghanem. Magic123: One image to high-quality 3d object generation using both 2d and 3d diffusion priors. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

[39] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya 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 *Proceedings of the 38th International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[40] Jiawei Ren, Liang Pan, Jiaxiang Tang, Chi Zhang, Ang Cao, Gang Zeng, and Ziwei Liu. Dreamgaussian4d: Generative 4d gaussian splatting. *arXiv preprint arXiv:2312.17142*, 2023.

[41] Stephen Roller, Sainbayar Sukhbaatar, Jason Weston, et al. Hash layers for large sparse models. *Advances in Neural Information Processing Systems*, 34:17555–17566, 2021.

[42] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bjrn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[43] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[44] 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. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022.

[45] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022.

[46] Qiuhong Shen, Xingyi Yang, and Xinchao Wang. Anything-3d: Towards single-view anything reconstruction in the wild. *arXiv preprint arXiv:2304.10261*, 2023.

[47] Qiuhong Shen, Xingyi Yang, Michael Bi Mi, and Xinchao Wang. Vista3d: Unravel the 3d darkside of a single image. In *European Conference on Computer Vision*, pages 405–421. Springer, 2024.

[48] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. *Advances in Neural Information Processing Systems*, 34:6087–6101, 2021.

[49] J Ryan Shue, Eric Ryan Chan, Ryan Po, Zachary Ankner, Jiajun Wu, and Gordon Wetzstein. 3d neural field generation using triplane diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20875–20886, 2023.

[50] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

[51] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

[52] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023.

[53] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. *arXiv preprint arXiv:2309.16653*, 2023.

[54] Matthias Teschner, Bruno Heidelberger, Matthias Müller, Danat Pomerantes, and Markus H Gross. Optimized spatial hashing for collision detection of deformable objects. In *Vmv*, pages 47–54, 2003.

[55] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A Yeh, and Greg Shakhnarovich. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12619–12629, 2023.

[56] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[57] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *Advances in Neural Information Processing Systems*, 36, 2024.

[58] Felix Wimbauer, Bichen Wu, Edgar Schoenfeld, Xiaoliang Dai, Ji Hou, Zijian He, Artsiom Sanakoyeu, Peizhao Zhang, Sam Tsai, Jonas Kohler, et al. Cache me if you can: Accelerating diffusion models through block caching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6211–6220, 2024.

[59] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. *Advances in neural information processing systems*, 29, 2016.

[60] Xiufeng Xie, Riccardo Gherardi, Zhihong Pan, and Stephen Huang. Hollownerf: Pruning hashgrid-based nerfs with trainable collision mitigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3480–3490, 2023.

[61] Yinghao Xu, Hao Tan, Fujun Luan, Sai Bi, Peng Wang, Jiahao Li, Zifan Shi, Kalyan Sunkavalli, Gordon Wetzstein, Zexiang Xu, and Kai Zhang. DMV3d: Denoising multi-view diffusion using 3d large reconstruction model. In *The Twelfth International Conference on Learning Representations*, 2024.

[62] Xingyi Yang, Daquan Zhou, Jiashi Feng, and Xinchao Wang. Diffusion probabilistic model made slim. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22552–22562, 2023.

[63] 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. *arXiv preprint arXiv:2310.08529*, 2023.

[64] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[65] Han Zhu, Mingsheng Long, Jianmin Wang, and Yue Cao. Deep hashing network for efficient similarity retrieval. In *Proceedings of the AAAI conference on Artificial Intelligence*, 2016.