Score Distillation via DDIM Inversion

by

Artem S. Lukoianov

B.S. Applied Math and Physics, MIPT, 2019
M.S. Data Science, EPFL, 2021

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2024

Authored by: Artem S. Lukoianov
Department of Electrical Engineering and Computer Science
May 17, 2024

Certified by: Justin Solomon
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Certified by: Vincent Sitzmann
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by: Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Score Distillation via DDIM Inversion

by

Artem S. Lukoianov

Submitted to the Department of Electrical Engineering and Computer Science
on May 17, 2024 in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

ABSTRACT

While 2D diffusion models generate realistic, high-detail images, 3D shape generation methods like Score Distillation Sampling (SDS) built on these 2D diffusion models produce cartoon-like, over-smoothed shapes. To help explain this discrepancy, in this paper we prove that the image guidance used in Score Distillation can be understood as the velocity field of a 2D denoising generative process, up to the choice of a noise term. In particular, after a change of variables, SDS resembles a high-variance version of Denoising Diffusion Implicit Models (DDIM) with a differently-sampled noise term: SDS introduces noise i.i.d. randomly at each step, while DDIM infers it from the previous noise predictions. This excessive variance can lead to over-smoothing and prevent the algorithm from generating realistic outputs. We show that a better noise approximation can be recovered by inverting DDIM in each SDS update step.This modification makes SDS's generative process for 2D images identical to DDIM, up to our change of variables. In 3D, it removes over-smoothing, preserves higher-frequency detail, and brings the generation quality closer to that of 2D samplers. Experimentally, our method achieves better or similar 3D generation quality compared to other works that improve SDS, all without training additional neural networks or 3D supervision. Our findings bridge the gap between 2D and 3D asset generation.

Thesis supervisor: Justin Solomon
Title: Professor of Electrical Engineering and Computer Science

Thesis supervisor: Vincent Sitzmann
Title: Professor of Electrical Engineering and Computer Science

Acknowledgments

I would like to thank my supervisor prof. Justin Solomon for the continuous guidance during my Master years at MIT, for his support of my research interests and a great example of how research should be done. Big thank you to my co-supervisor prof. Vincent Sitzmann with whom we regularly have fruitful brainstorming sessions, and who helps my research stay on track with a valuable feedback. I am grateful to the entire Meta Reality Labs research team in Pittsburgh, PA. My internship there served provided me with a basis of my understanding of the field, and in particular, to Timur Bagautdinov, my internship mentor and a friend, with whom we kept regular discussion of the project. Thank you to Haitz Sáez de Ocáriz Borde for the productive conversations and the help with experimental setup of this work. Furthermore, I would like to thank Kristjan Greenewald, Vitor Campagnolo Guizilini and Lingxiao Li for valuable advise during the work on this project. Finally, my biggest thanks to my family, friends and close people, who unconditionally supported me in the hardest times.

The MIT Geometric Data Processing group acknowledges the generous support of the Toyota–CSAIL Joint Research Center.

Contents

Chapter 1

Introduction

Image generative modeling saw drastic quality improvement with the advent of text-to-image diffusion models [1] trained on billion-scale datasets [2] and large parameter counts [3]. From a short text prompt, these models generate photorealistic images, with strong zero-shot generalization to new classes of objects [4]. Efficient training methods for image data, combined with Internet-scale datasets, enabled the development of these generalizable models. However, applying similar techniques to domains where huge datasets are scarce, such as 3D shape generation, remains challenging.

The coherency and photorealism of images produced by state-of-the-art 2D generative models, however, suggests that these models do understand some 3D structure, motivating methods like Score Distillation Sampling (SDS) [5], [6], which optimize 3D representations (e.g., NeRFs [7], InstantNGPs [8] or Gaussian Splattings [9], [10]) using queries to a 2D generative model [11]. In every iteration, SDS renders the current state of the 3D representation from a random viewpoint, adds noise to the result, and then denoises it using the pre-trained 2D diffusion model conditioned on a prompt. The difference between the added and predicted noise is used as a gradient-style update on the rendered images, which is propagated to the parameters of the 3D model. The underlying 3D representation helps make the generated images view-consistent, and the 2D model guides individual views towards a learned distribution of realistic images.

In practice, however, as was noted in previous works [12]–[14], SDS often produces 3D representations with over-saturated colors and over-smoothed textures (see fig. 2.1 d), not matching the quality of the underlying 2D model. Existing approaches tackling this problem improve quality at the cost of expensive re-training or fine-tuning of the image diffusion model [12], complex multi-stage handling of 3D representations, like mesh extraction and texture fine-tuning [12], [15]–[17] , or tweaking the SDS guidance [13], [14].

As an alternative to engineering-based solutions to the challenges with SDS, in this paper we re-analyze the vanilla SDS algorithm to understand the underlying source of these artifacts. Our key insight is that the SDS update rule can be seen as a step along an approximation of the DDIM velocity field. In particular, we derive Score Distillation from DDIM with a change of variables to the space of single-step denoised images. Under our change of variables, SDS updates are nearly identical to DDIM updates, apart from one difference: while DDIM samples noise *conditionally* on the predictions in the previous timestep, SDS resamples random noise i.i.d. in every iteration. This resampling breaks the denoising trajectory for

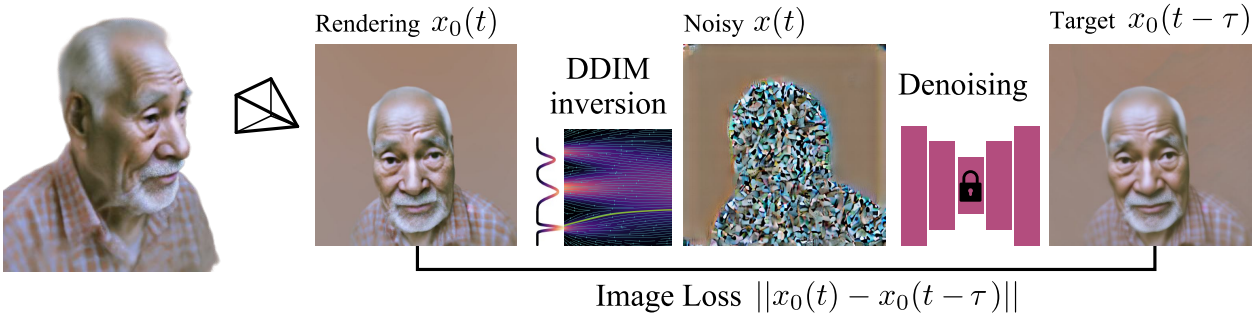

Figure 1.1: **Overview.** In each training iteration, we render a random view of the current 3D shape, run DDIM inversion up to the current noise level t, and denoise the image with a pre-trained diffusion model for noise level $t - \tau$. We optimize the 3D shape with the denoised image as a target for the rendered view.

each independent view and introduces excessive variance. Our perspective unifies DDIM and SDS and helps explain why SDS can produce blurry and over-saturated results: the variance-boosting effect of noisy guidance is usually mitigated with high CFG values to reduce sample diversity at the cost of over-saturation [18].

Based on our analysis, we propose an alternative score distillation algorithm, closing the gap to DDIM. In particular, we regress the conditional noise sample required for consistency of the denoising trajectories by inverting DDIM on each step of score distillation (fig. 1.1). Empirically, this modification yields 3D objects with high-quality colors and textures consistent with the underlying 2D diffusion model, without any expensive fine-tuning or modifications to the neural network. Moreover, for 2D generation, our method closely approximates DDIM for image generation while preserving the iterative, incremental generation schedule of SDS (fig. 2.1).

Our key contributions are as follows:

- We prove that guidance for each view in the SDS algorithm is a simplified re-parameterization of DDIM sampling: vanilla SDS samples random noise at each step, while DDIM keeps the trajectories consistent with previously-predicted noise.
- We show that replacing the problematic random sampling in SDS with a DDIM inversion process improves generation quality, closing the quality gap to samples from the 2D model and demonstrating competitive performance to state-of-the-art 3D generation.
- In a systematic evaluation study we compare our method with current state-of-the-art score distillation methods both qualitatively and quantitatively and show that it achieves similar or better generation quality, all while not requiring training additional neural netowrks, or multiple stages of generation.

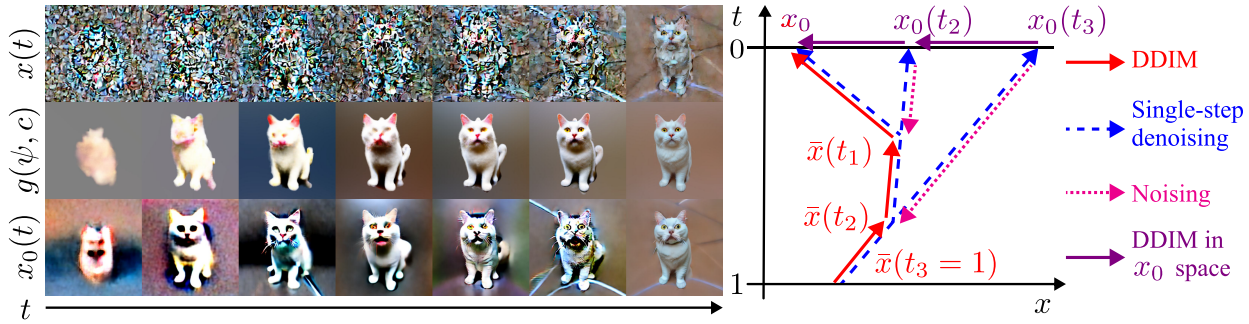

Figure 1.2: Left: Evolution of different variables of Score Distillation in time. On the top row we depict how noised images $x(t)$ evolve in 2D generation, in the middle we present the evolution of a NeRF in 3D generation, and in the bottom we show how the single step denoised variable $x_0(t)$ changes with t. Right: Each step of DDIM is a short step towards single step denoised image. This can be seen as a step to $x_0(t)$ and a step back to the noised image. We reorder the vectors in this process to obtain a process on $x_0(t)$.

Chapter 2

Related work

Figure 2.1: Score Distillation Sampling (SDS) builds 3D shapes from 2D images generated by a diffusion model like DDIM. While DDIM produces high-quality images (a, CFG=7.5), the same diffusion model, when used to generate a 2D image with SDS, yields a blurry result (b, CFG=7.5); in 3D, the method yields over-saturated and simplified shapes (d, CFG=100). By replacing the noise term in SDS to agree with DDIM in the 2D case, our algorithm better matches the quality of the diffusion model in 2D (c, CFG=7.5) and significantly improves 3D generation (e, CFG=7.5).

Distilling 2D generative models into 3D assets. Score Distillation was first introduced by works like Dreamfusion (SDS) [5], Score Jacobian Chaining [6], and Magic3D [15]. The key idea is to use a frozen diffusion model trained only on 2D images and "distill" it into 3D assets. A volumetric representation of the current shape is rendered from a random view, perturbed with a random noise sample, and denoised with the diffusion model; then, the difference between added and predicted noise is used as guidance to improve the rendering. These works, however, suffer from over-smoothing and lack of detail. Usually a high value of classifier free guidance (CFG) [19] is used to reduce the variance of the results, at the cost of over-saturation [18].

ProlificDreamer [12] generates sharp, detailed results with standard CFG values and without over-saturation. The key is to overfit a second diffusion model to specifically denoise the current 3D shape. Fine-tuning the second model, however, is cumbersome and requires additional hyperparameters. It also remains unclear why this change should be effective theoretically. Some further improve on ProlificDreamer's results or try to explain its behavior. SteinDreamer [20], for example, hypothesizes that ProlificDreamer's improvements come from variance reduction of the sampling procedure.

Other papers propose heuristics that improve SDS pipelines. For example, [13], [21] decompose the guidance terms and speculate about their relative importance. Empirically, visual quality can be improved by suppressing the denoising term with negative prompts [13] or highlighting the classification term [21]. Other works [12], [15]–[17] use multi-stage optimization: they first train a volumetric representation and then extract a mesh or voxel grid to fine-tune geometry and texture. HiFA [14] uses a pretrained mono-depth estimation network, time annealing, supervision in latent space, kernel smoothing, and z-variance regularization to improve quality of single-stage NeRF generation.

Rather than improving SDS by augmenting its pipeline, in this work we prove that Score Distillation is a high-variance version of a 2D denoising process, suggesting a simple modification to the SDS formulas that significantly improves 3D generation.

3D generation by training on multi-view data. Recent advancements in 3D generation leverage multi-view data or training on 3D data. Zero123 [22] and MVDream [23] introduce diffusion models that generate consistent multi-view images from text. A 3D radiance field is then obtained via score distillation. Video generative models can be fine-tuned on videos of camera tracks around 3D objects, similarly yielding a model that samples multi-view consistent images that can be used to train a 3D radience field [24], [25]. Diffusion with Forward Models [26] and Viewset Diffusion [27] directly train a 3D generative model from 2D image observations.

While these methods excel at generating multi-view consistent, plausible 3D objects, they depend on multi-view data with known camera trajectories, limiting them to synthetic or small bundle-adjusted 3D datasets. We instead focus on methods that require only single-view training images.

Chapter 3

Background

Diffusion models. Denoising Diffusion Implicit Models (DDIM) generate images using diffusion [28]–[30]. After training the denoiser ϵ_θ^t and freezing its weights θ, the denoising process can be seen as an ODE on rescaled noisy images $\bar{x}(t) = x(t)/\sqrt{\alpha(t)}$. For a prompt y and current time step $t \in [0, 1]$, the denoising process satisfies:

$$\frac{d\bar{x}(t)}{dt} = \epsilon_\theta^t\big(\sqrt{\alpha(t)}\bar{x}(t), y\big)\frac{d\sigma(t)}{dt}, \tag{3.1}$$

where $\bar{x}(1)$ is sampled from a Gaussian distribution, $\sigma(t) = \sqrt{1-\alpha(t)}/\sqrt{\alpha(t)}$, and $\alpha(t)$ are scaling factors. When discretized with forward Euler, this equation yields the following update to transition from step t to a less noisy step $t - \tau < t$:

$$\bar{x}(t-\tau) = \bar{x}(t) + \epsilon_\theta^t\big(\sqrt{\alpha(t)}\bar{x}(t), y\big)\left[\sigma(t-\tau) - \sigma(t)\right]. \tag{3.2}$$

The DDIM update ODE can also be integrated in reverse direction to estimate $\bar{x}(t)$ for any $t \in [0, 1]$ from a clean image $x_0 = \bar{x}(0)$.

Classifier-free guidance. Classifier-free guidance (CFG) [19] provides high-quality conditional samples without gradients from auxilary models [31]. CFG modifies the noise prediction $\hat{\epsilon}_\theta^t$ (score function) by linearly combining conditional and unconditional predictions:

$$\epsilon_\theta^t\big(x(t), y\big) = \hat{\epsilon}_\theta^t(x(t), \varnothing) + \gamma \cdot \big(\hat{\epsilon}_\theta^t(x(t), y) - \hat{\epsilon}_\theta^t(x(t), \varnothing)\big)\,, \tag{3.3}$$

where the guidance scale γ is a scalar, with $\gamma = 0$ corresponding to unconditional sampling and $\gamma = 1$ to conditional. In practice, larger values $\gamma > 1$ are necessary to obtain high-quality samples at a cost of reduced diversity and extreme over-saturation (fig. 3.1). For the rest of the paper we will be using the modified CFG version of the denoiser $\epsilon_\theta^t\big(x(t), y\big)$.

Score distillation. Diffusion models efficiently generate images and can learn to represent common objects from arbitrary angles [32] and with varying lighting [5] when trained on large datasets. Capitalizing on this success, Score Distillation Sampling (SDS) [5] distills a pre-trained and fixed diffusion model ϵ_θ^t to produce a 3D asset. In practice, the 3D shape is usually parameterized by a NeRF [7], InstantNGP [8] or Gaussian Splatting [9]. Multiple works additionally extract an explicit representation for further optimization [12], [15]–[17]. We use InstantNGP [8] to balance between speed and ease of optimization.

Prompt – "colored photograph of an old man"

Figure 3.1: The effect of different CFG values on the quality of 2D generation via StableDiffusion 2.1 [1]. For the small values, the model tends to ignore certain words in the prompt. For high values, the images get over-saturated.

Denote the parameters of a differentiable 3D shape representation by $\psi \in \mathbb{R}^d$ and differentiable rendering by a function $g(\psi, c) : \mathbb{R}^d \times C \to \mathbb{R}^{N \times N}$ that returns an image given camera parameters $c \in C$. Intuitively, in each iteration, SDS samples c, renders the corresponding (image) view $g(\psi, c)$, perturbs it with $\epsilon \sim \mathcal{N}(0, I)$ to level $t \sim [0, 1]$, and denoises it with ϵ_θ^t; the difference between the true and predicted noise is propagated to the parameters of the volume. More formally, after sampling the camera view c and randomly drawing a time t, SDS renders the volume and adds Gaussian noise ϵ to obtain a noisy image

$$x(t) = \sqrt{\alpha(t)} g(\psi, c) + \sqrt{1 - \alpha(t)} \epsilon.$$

Then, SDS updates the volume by using the following gradient(-like) direction to update its parameters ψ:

$$\nabla_\psi \mathcal{L}_{SDS} = \mathbb{E}_{t,\epsilon,c} \sigma(t) \left[\epsilon_\theta^t \big(x(t), y \big) - \epsilon \right] \frac{\partial g}{\partial \psi}. \tag{3.4}$$

This update rule may not correspond to the true gradient of a function; there are many hypotheses about its effectiveness [5], [13], [20], [21]. In this work, we prove that the above process can be seen as a high-variance version of DDIM after a change of variables.

Chapter 4

Linking SDS to DDIM

4.1 Discrepancy in image sampling.

Beyond the lack of formal justification of eq. (3.4), in practice SDS results are over-saturated and miss detail for high CFG values or are blurry for low CFG values. To illustrate this phenomenon, fig. 2.1 shows a simple experiment, inspired by [12]: We replace the volumetric representation in eq. (3.4) with an image $g_{2D}(\psi_{2D}, c) := \psi_{2D} \in \mathbb{R}^{N \times N}$. In this case, SDS becomes an image generation algorithm comparable to other sampling algorithms like DDIM [33]. Even in this 2D setting, SDS fails to generate sharp details, while DDIM with the same underlying diffusion model produces photorealistic results, motivating our derivation below.

4.2 Why not use DDIM as guidance?

Given the experiment above, a natural question to ask is if it is possible to directly combine DDIM's update direction from eq. (3.1) with the SDS guidance in eq. (3.4) to update the 3D representation. The problem with this approach lies in the discrepancy between the training data of the denoising model and the images generated by rendering the current 3D representation. More specifically, the denoising network expects an image with a certain level of noise corresponding to time t as defined by the forward (noising) diffusion process, whereas renderings of 3D representations $g(\psi, c)$ evolve from a blurry cloud to a well-defined sample (fig. 1.2).

4.3 Evolution of $x_0(t)$.

Instead of seeing DDIM as a denoising process defined on noisy images $x(t)$, we consider it as a process defined on a different variable:

$$x_0(t) = \bar{x}(t) - \sigma(t)\epsilon_\theta^t\big(x(t), y\big). \tag{4.1}$$

In words, $x_0(t)$ is the noisy image at time t denoised with a single step of noise prediction. Empirically, the evolution of $x_0(t)$ is similar to the evolution of $g(\psi, c)$—from a blurry sample

Figure 4.1: Examples of multiple views of 3D objects generated by from our model.

to a well-defined, sharp sample. We visually compare the two processes in fig. 1.2. This similarity motivates us to rewrite eq. (3.1) in terms of $x_0(t)$, and to understand SDS as applying similar updates to the 3D representation. Thus we demonstrate that SDS guidance approximates DDIM's evolution of $x_0(t)$. In particular, the velocity field determining the time evolution of $x_0(t)$ images under DDIM gets used as a gradient for each rendering of the volumetric representation in SDS.

4.4 Reparametrizing DDIM.

Figure 1.2 shows schematically how the DDIM update—defined on $\bar{x}(t)$—alternates between denoising to obtain $x_0(t)$ and adding noise to get a cleaner image $\bar{x}(t-\tau)$. We can rearrange the equations to reorder the steps, first adding noise to $x_0(t)$ and then denoising it to estimate $x_0(t-\tau)$.

Consider neighboring time points t and $t-\tau < t$ in discretized DDIM eq. (3.2) (lower time corresponds to less noise). We rewrite eq. (3.2) using the definition of $x_0(t)$ from eq. (4.1) to find

$$x_0(t-\tau) = x_0(t) - \sigma(t-\tau)\left[\epsilon_\theta^{t-\tau}\big(x(t-\tau), y\big) - \epsilon_\theta^{t}\big(x(t), y\big)\right], \tag{4.2}$$

which is consistent with the intuition behind SDS: improving an image involves perturbing the current image and then denoising it with a better noise estimate. We cannot directly

apply eq. (4.2) to SDS in the 3D case, since it still depends on $x(t)$; if we think of $x_0(t)$ as similar to a rendering of the current 3D representation from a single camera angle, it is unclear how to obtain a consistent set of preimages $x(t)$ at each step of 3D generation. From eq. (4.1), however, $x(t)$ should satisfy the following fixed point equation:

$$x(t) = \sqrt{\alpha(t)}x_0(t) + \sqrt{1-\alpha(t)}\epsilon_\theta^t\big(x(t), y\big), \tag{4.3}$$

or rewritten in terms of noise $\epsilon = [x(t) - \sqrt{\alpha(t)}x_0(t)]/\sqrt{1-\alpha(t)}$:

$$\epsilon = \epsilon_\theta^t\big(\sqrt{\alpha}x_0(t) + \sqrt{1-\alpha(t)}\epsilon, y\big). \tag{4.4}$$

Define $\kappa^t\big(x_0(t)\big) = \epsilon$ as a solution of this equation given $x_0(t)$. Then, we can write:

$$\begin{aligned} &\epsilon_\theta^t\big(x(t), y\big) = \kappa^t\big(x_0(t)\big) \\ &x(t-\tau) = \sqrt{\alpha(t-\tau)}x_0(t) + \sqrt{1-\alpha(t-\tau)}\kappa^t\big(x_0(t)\big). \end{aligned} \tag{4.5}$$

Thus eq. (4.2) turns into:

$$x_0(t-\tau) = x_0(t) - \sigma(t-\tau)\Big[\underbrace{\epsilon_\theta^{t-\tau}\Big(\overbrace{\sqrt{\alpha(t-\tau)}x_0(t) + \sqrt{1-\alpha(t-\tau)}\kappa^t\big(x_0(t)\big)}^{x_0 \text{ noised with } \kappa^t \text{ to } t-\tau}, y\Big)}_{\text{predicted noise}} - \underbrace{\kappa^t\big(x_0(t)\big)}_{\text{noise sample } \kappa^t}\Big]. \tag{4.6}$$

If one had access to $\kappa^t\big(x_0(t)\big)$, eq. (4.6) would turn into a discrete update rule that simulates DDIM and directs images toward the conditional distribution learnt by the diffusion model. Moreover, we can already see that the structure of eq. (4.6) is very similar to the SDS update rule in eq. (3.4). Note that the update direction in eq. (4.6) is the same as in the SDS update rule in eq. (3.4), whereas κ^t plays the role of the random noise sample ϵ. We could use it as a guidance for the 3D generative process in SDS by replacing ϵ in eq. (3.4) with $\kappa^t(x_0(t))$. In practice, however, it is hard to solve eq. (4.4), as ϵ_θ^t is high-dimensional and nonlinear. Below we show that a naïve approximation replacing κ^t with a Gaussian yields SDS, and we will propose alternatives that are more faithful to the derivation above.

4.5 Continuous perspective.

Our derivation above manipulates the time-stepping procedure of DDIM, but a similar argument applies to the ODE version of the method. One can obtain an alternative form of eq. (4.6) by reparametrizing the ODE eq. (3.1) as:

$$\frac{dx_0(t)}{dt} = -\sigma(t)\frac{d}{dt}\epsilon_\theta^{(t)}\left(x_0(t) + \sigma(t)\kappa^t(x_0(t)), y\right). \tag{4.7}$$

Details are provided in the appendix A.2. This ODE is a velocity field that projects a starting image to a conditional distribution learned by the diffusion model. Discretizing this equation leads to eq. (4.6).

4.6 SDS as a special case.

From eq. (4.6), to get a cleaner image, we need to bring the current image to time t with noise sample κ^t, denoise the obtained image, and then subtract the difference between added and predicted noise from the initial image. A coarse approximation of κ uses i.i.d. random noise $\kappa^t_{SDS}(x_0(t)) \sim \mathcal{N}(0, I)$, matching the forward process by which diffusion adds noise. This choice of κ^t_{SDS} precisely matches the update rule eq. (4.6) to the SDS guidance in eq. (3.4).

Chapter 5

Guiding 3D Generation with Ancestral Sampling

5.1 DDIM Inversion

As we have shown, SDS follows the velocity field of reparametrized DDIM in eq. (4.6), when $\kappa^t(x_0(t))$ is randomly sampled in each step. Our derivation, however, suggests that $\kappa^t_{SDS}(x_0(t))$ could be improved by bringing it closer to a solution of the fixed-point equation in eq. (4.4). Indeed, randomly sampling κ^t_{SDS} as in Dreamfusion yields excessive variance and blurry results for standard CFG values, while using higher CFG values leads to over-saturation and lack of detail. On the other hand, solving eq. (4.6) exactly is challenging due to its high dimensionality and nonlinear nature.

To improve our prediction of κ^t we propose to invert DDIM by integrating its ODE in eq. (3.1) in reverse, that is, solving the ODE with the t direction evolving backwards relative to eq. (3.1) (from images to noise). This process approximates but is not identical to the exact solution of kappa: the fixed-point solution of eq. (4.6) attempts to invert a single large step of DDIM, while running the ODE in reverse inverts the entire DDIM trajectory. In section 6.2 we ablate other choices for κ^t and conclude that inverting DDIM offers the best approximation quality among the studied alternatives.

As we can see, in eq. (4.6), the added noise inverts eq. (4.4) up to noise level t, however, the denoising step happens in slightly lower time $t-\tau$. Intuitively τ controls the effective step size in the denoising (or image improvement) process. To accommodate this, we maintain a global time variable t that linearly decays for all the views. Then, on each update step we run DDIM inversion up to $t+\tau$. Here τ is a small constant that regulates the size of the denoising step in eq. (4.6). In practice, we did not find the algorithm to be particularly sensitive to this constant.

Figure 5.1 shows the effect of inferring the noise via DDIM inversion instead of sampling it randomly. The special structure of the improved κ^t results in more consistent single-step generations and produces intricate features of the rendering at much earlier times. When inverted and not randomly sampled, the noise appears 'in the right place' of the volume: in SDS the noise covers the whole view, including the background, whereas in ours the noise is concentrated on the meaningful part of the 3D shape. This leads to more geometrically

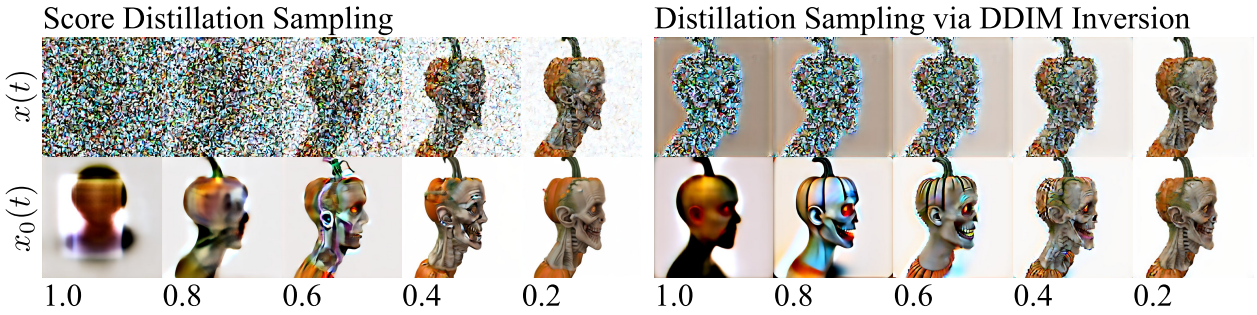

Figure 5.1: Comparison of intermediate variables in SDS and Ours for different timesteps t. Starting with a rendering of a 3D shape we demonstrate how each algorithm perturbs it ($x(t)$ variable on the top row) and how it is denoised with a single step of diffusion ($x_0(t)$ variable on the bottom row). The prompt used is "Pumpkin head zombie, skinny, highly detailed, photorealistic, side view".

and time coherent $x_0(t)$ predictions even for large t values, which is not the case for SDS. The reduced variance drastically increases sharpness and the level detail of the generations. Moreover, it allows to reduce CFG value of generation γ_{fwd} to the standard 7.5 and thus avoid over-saturation. Another interesting finding in our work is that DDIM inversion works the best, when the reverse integration is performed with negative CFG $\gamma_{\text{inv}} = -\gamma_{\text{fwd}} = -7.5$. We perform a detailed ablation study of the CFG values in section 6.2.

Algorithm 1 Original Dreamfusion algorithm

procedure DREAMFUSION(y)
  for i in range(n_iters) **do**
    $t \leftarrow \text{Uniform}(0, 1)$
    $c \leftarrow \text{Uniform}(\mathcal{C})$
    $\epsilon \leftarrow \text{Normal}(0, I)$
    $x_t \leftarrow \sqrt{\alpha(t)} g(\psi, c) + \sqrt{1 - \alpha(t)}\epsilon$
    $\nabla_\psi \mathcal{L}_{SDS} = \sigma(t) \left[\epsilon_\theta^{(t)}(x_t, y) - \epsilon\right] \frac{\partial g}{\partial \psi}$
    Backpropagate $\nabla_\psi \mathcal{L}_{SDS}$
    SGD update on ψ
  end for
end procedure

Algorithm 2 Suggested algorithm

procedure OURS(y)
  for i in range(n_iters) **do**
    $t \leftarrow 1 - i/n\_iters$
    $c \leftarrow \text{Uniform}(\mathcal{C})$
    $\epsilon \leftarrow \kappa^{t+\tau}(x_0(t))$
    $x_t \leftarrow \sqrt{\alpha(t)} g(\psi, c) + \sqrt{1 - \alpha(t)}\epsilon$
    $\nabla_\psi \mathcal{L}_{SDS} = \sigma(t) \left[\epsilon_\theta^{(t)}(x_t, y) - \epsilon\right] \frac{\partial g}{\partial \psi}$
    Backpropagate $\nabla_\psi \mathcal{L}_{SDS}$
    SGD update on ψ
  end for
end procedure

Figure 5.2: Comparison of the original SDS algorithm and our proposed changes.

The overview of our method is presented in fig. 1.1. We provide more details about the inversion algorithm in section 6.2.

5.2 Implementation Details

Geometry regularization. Reducing the variance of the guidance allows us to use smaller CFG values. We use $\gamma = 7.5$, which is standard for 2D generation. As in fig. 3.1, when operating with smaller CFG values, the diffusion model tends to ignore certain parts of the prompts, like "colored" for $\gamma < 10$ in this particular case.

Dreamfusion [23] augments prompts with information about the direction of the view to avoid the *Janus problem*, wherein the model tends to produce frontal views all around the shape due to its bias for these views, as they were more dominant in the training data. Since our algorithm allows to reduce the CFG to the standard value of 7.5, the diffusion model begins to ignore the view augmentation prompting =due to the aforementioned phenomenon, yielding a stronger Janus problem compared to Dreamfusion; we see the same behavior for other baselines that reduce CFG. To tackle this problem, we use Perpendicular Negative Prompting [32] and add a small entropy term $\sim \mathcal{N}(0, 0.3\sigma(t)I)$ to the inverted noise to reduce mode-seeking behavior as in [34].

Another common issue in Score Distillation is the *hollow face illusion* [35] (fig. 5.3). To address this, we add a simple loss favoring convex shapes over concave ones. In particular, we use a normal map extracted from the current volume as described in the original SDS work [5]. We calculate the sine between adjacent normals (moving from left to right and from top to bottom) and penalize it to be as negative as possible. We activate this loss for the first 40% of the training steps with weight $\alpha_{\text{convexity}} = 0.1$ to break the symmetry.

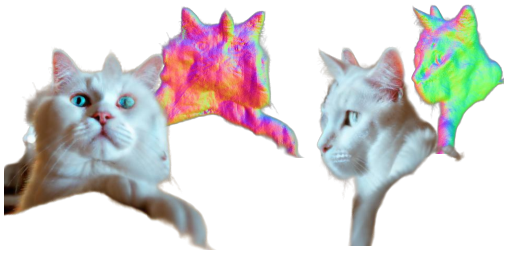

Figure 5.3: In the hollow face illusion, generated views tend to be concave or flat. We combat this behavior with a simple convexity loss that biases the generation towards convex shapes.

System details. We implement our algorithm in `threestudio` [36] on top of SDS [5]. We use Stable Diffusion 2.1 [1] as the underlying diffusion model. For volumetric representation we follow the implementation of Dreamfusion and use InstantNGP [8]. Instead of randomly sampling time t as in SDS, we maintain a global parameter t that linearly decays from 1 to 0.2 (we found that lower time values do not significantly contibute to the generation quality). Next, for each step renders a 512×512 random view and infers $\kappa^{t+\tau}$ by running DDIM inversion for $\texttt{int}(10t)$ steps. We sample $\tau \sim U(0, \frac{1}{30})$. We use NVIDIA A6000 GPUs and run each generation for $10k$ steps with learning rate of 10^{-2}, which takes approximately 2 wall-clock hours per shape generation.

Chapter 6

Experiments

6.1 3D Generation

We demonstrate the high-fidelity 3D shapes generated with our algorithm in fig. 4.1.

Qualitative comparisons. For evaluation we adopt a protocol similar to previous works [12], [13]. We compare 3D generation quality with the results reported in their respective works in fig. 6.1. For the baselines we chose: Dreamfusion [5] as the work we build on top of, Noise Free Score Distillation [13] that suggests to use negative prompts in SDS, ProlificDreamer [12] that adopts additional fine-tuning stages and trains a separate neural network to denoise the current 3D shape, and HiFA [14] that suggests a series of improvements such as additional supervision in the latent space, NeRF regularization, and depth supervision with a separate mono-depth estimation network. As can be seen from the figures, relating SDS with ancestral sampling by inverting DDIM allows to achieve similar or better results compared with other state-of-the-art methods.

Quantitative comparison. We follow previous works [5], [20], [21] to quantitatively evaluate generation quality of our algorithm. In table 6.1 we provide the CLIP score [37] to measure prompt-generation alignment. We use the `torchmetrics` [38] package and the ViT-B/16 model [39]. We also include CLIP Image Quality Assessment (IQA) [40] to measure quality ("Good photo." vs "Bad photo."), sharpness ("Sharp photo." vs "Blurry photo."), and realism ("Real photo." vs "Abstract photo.") of the generations. For each method we run generations on 43 prompts and sample 50 views around the generated objects. For baselines that include multiple stages we run only the first one for a fair comparison with ours. In the table we also report the percentage of generations that run out-of-memory or generate an empty volume as 'Diverged'. Additionally we provide average run time and VRAM usage for readers' reference. For the VRAM column we at first take the maximum usage of the GPU's memory within each run and then average the obtained values. As a lot of baselines are not open-source, we use their implementation in threestudio [36] for all the baselines. Our approach significantly outperforms SDS and shows better or similar results compared to other state-of-the-art methods, while offering a simple fix to SDS and not requiring additional supervision or multiple stages of training.

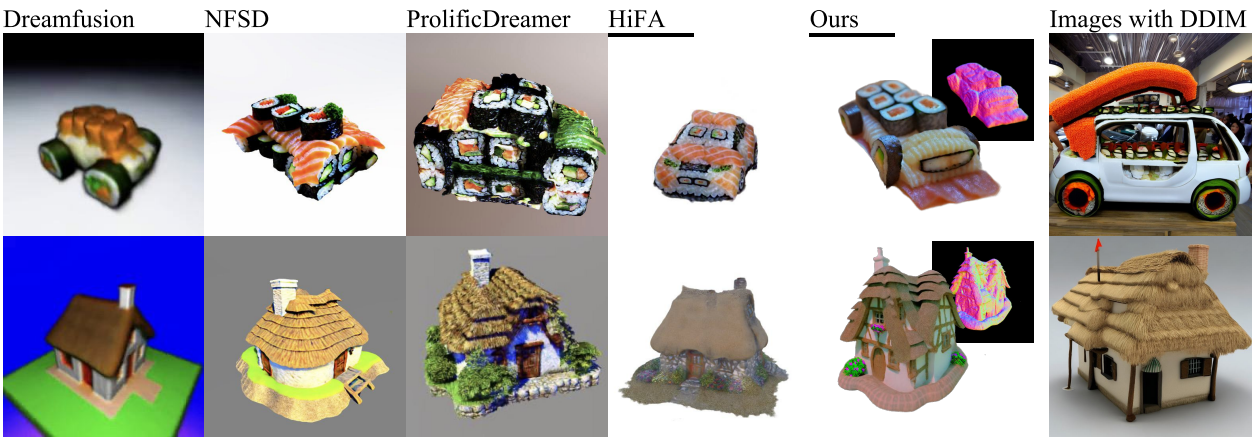

Figure 6.1: Comparison of 3D generation with other methods using reported results from the respective papers. The prompts are "A 3D model of an adorable cottage with a thatched roof" and "A car made out of sushi". On the right side we provide 2D images generated with the same diffusion model via DDIM for reference.

Table 6.1: Quantitative comparisons to baselines for text-to-3D generation, evaluated by CLIP Score, CLIP IQA and FID metrics. We report scores based on mean and standard deviation across 50 views

Method	CLIP Score (↑)	CLIP IQA (%) ↑			Diverged (%) ↓	Time	VRAM
		"quality"	"sharpness"	"real"			
SDS [5], $10k$ steps	29.81 ± 2.49	76 ± 6.6	99 ± 1.2	98 ± 2.4	18.6	66min	6.2GB
VSD [12], $25k$ steps	33.31 ± 2.39	77 ± 6.7	98 ± 1.3	96 ± 4.4	23.2	334min	47.9GB
ESD [34], $25k$ steps	32.79 ± 2.15	77 ± 7.2	98 ± 1.2	97 ± 2.7	14.0	331min	46.8GB
HIFA [14], $25k$ steps	32.80 ± 2.35	81 ± 6.5	98 ± 1.5	97 ± 1.2	4.7	235min	46.4GB
Ours, $10k$ **steps**	33.47 ± 2.49	82 ± 6.3	98 ± 1.3	97 ± 1.2	4.7	119min	39.2GB

6.2 Ablation

Proposed improvements. Figure 6.2 provides the ablation of the major proposed improvements. Starting from the setting of Dreamfusion [5] with CFG 7.5 we step by step add: increased resolution of NeRF rendering (from 64×64 to 512×512), annealing linear schedule on t, and, our core contribution – inference of the noise with DDIM inversion. The results clearly demonstrate that the main difference in quality comes from the inferred noise sample, while other improvements unlock its full potential.

Choice of $\kappa^t(x)$. The key component of our algorithm is the choice of the function $\kappa^t(x)$ that infers the noise sample. In theory, $\kappa^t(x)$ should be a solution of eq. (4.4), however, in practice, it is hard to derive its exact solution. In this part we study different choices of $\kappa^t(x)$ and analyze the numerical error of each choice in eq. (4.4). We consider the following choices:

1. **Random, re-sampled** - $\kappa^t(x)$ is noise randomly sampled on each new update step from $\mathcal{N}(0, I)$;

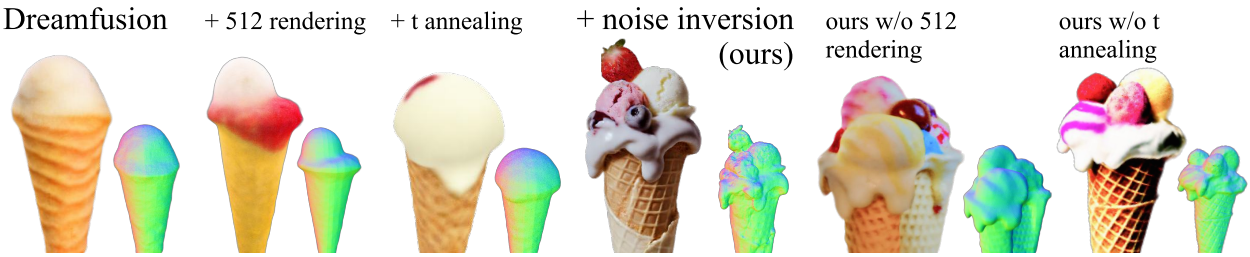

Figure 6.2: Ablation study of the proposed improvements

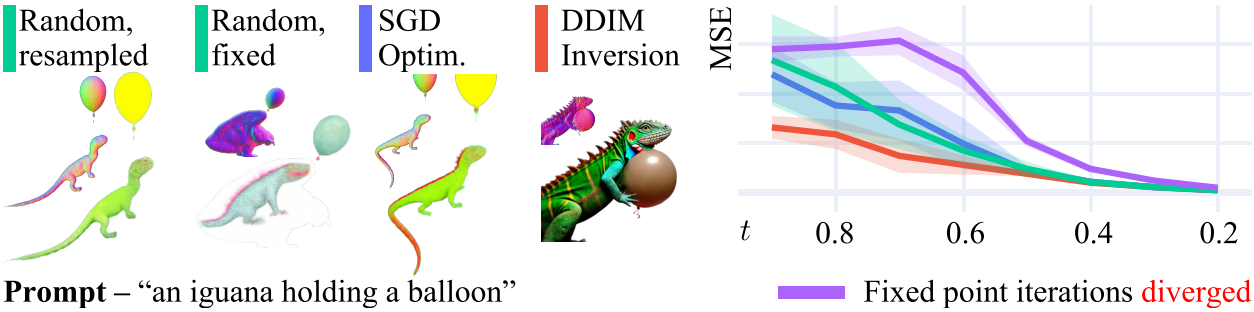

Figure 6.3: The effect of different $\kappa^t(x_0(t))$ choices on 3D generation quality.

2. **Random, fixed** - $\kappa^t(x)$ is noise randomly sampled from $\mathcal{N}(0, I)$ once and fixed for each iteration;

3. **Fixed point iteration** - since the optimal solution is a fixed point of eq. (4.4), we initialize $\kappa^t(x)$ to a random noise sample and run the fixed point iteration algorithm [41] for 10 steps. In our experience, increasing the number of steps up to 200 does not improve the reults.

4. **SGD optimization** - we attempt to regress $\kappa^t(x)$ via gradient decent for 10 steps, also initializing it to a random noise sample at the beginning.

5. **DDIM inversion** - finally, we run DDIM inversion for $10t$ (i.e. we need less steps for smaller t) steps to time t. We use negative CFG $\gamma_{\text{inversion}} = -7.5$ for inversion and positive $\gamma_{\text{forward}} = 7.5$ for forward inference.

We analyze all of the choices in terms of both, qualitative results in 3D generation presented on the left side of fig. 6.3, and the error induced in fig. 6.3 (re-scaled to x_0 variable due to its ambiguity around 0), presented on the right side of fig. 6.3. Since re-sampled and fixed noises produce the same erorr in eq. (4.4), they are represented with the same line on the plot for clarity.

As can be seen, the regressed noise has a big impact on the final generations. Both Fixed Point Iterations algorithms, and optimization via gradient descent fail to improve the approximation of κ^t, while Fixed Point Iteration diverges. We speculate that this is due to the high-dimensional and highly non-linear nature of the denoiser. On the other hand-side DDIM inversion yields a reasonable approximation of κ^t and significantly improves 3D generation.

CFG value for inversion. Multiple works [42], [43] have reported that DDIM inversion accumulates big numerical error for CFG value $\gamma > 1$. One of the interesting findings in this

Figure 6.4: Comparison of different DDIM inversion strategies in 2D. Note that here we compare the quality of inversion, i.e. the forward pass is the entire DDIM trajectory.

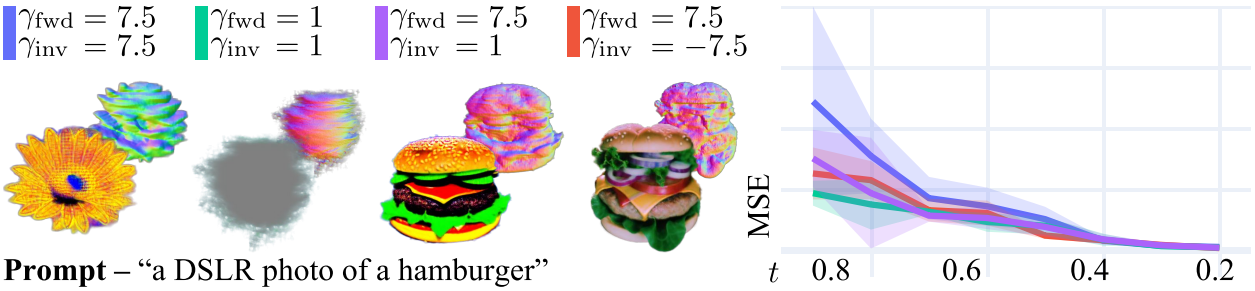

Figure 6.5: Comparison of different DDIM inversion strategies. On the left we demonstrate generation results for different strategies of using CFG values for denoising and inversion. On the right side we provide a numerical error in eq. (4.4) enduced by the inferred noise.

work is that DDIM inversion for CFG $\gamma_{\text{fwd}} > 1$ can be adequately estimated by running the inversion with negative CFG value $\gamma_{\text{inv}} = -\gamma_{\text{fwd}}$. As can be seen from fig. 6.4, when inverting the entire DDIM trajectory, starting with a clean image, the best inversion strategy is to use $\gamma_{\text{inv}} = 1$ and then regenerate the image by running DDIM trajectory with $\gamma_{\text{fwd}} = 1$. This approach was suggested in [43] and inverts the image almost perfectly. Other approaches introduce bias and derive only an approximation of the image on the forward pass.

In 3D, however, the story is different. In fig. 6.5 we compare different inversion strategies both qualitatively and quantitatively. We can see that in terms of the numerical error, the naive approach of $\gamma_{\text{inv}} = \gamma_{\text{fwd}} = 7.5$ yields the biggest bias, while the rest of the strategies perform on par. As we can see from the qualitative results on the left, $\gamma_{\text{inv}} = \gamma_{\text{fwd}} = 7.5$ introduces too much numerical error and the generation process drifts to a random direction. The best performing for inversion, combination of $\gamma_{\text{inv}} = \gamma_{\text{fwd}} = 1$ does not work for the generation task. We explain this behaviour by the fact that in our previous experiment, the original image in 2D already contains its class information. Inverting it with $\gamma_{\text{inv}} = 1$ encodes this class information into the noise and thus it does not need additional CFG on the reconstruction step. In generation, however, the starting shape of the NeRF does not contain any class information and thus needs bigger CFG on the forward pass. Introducing it only on the forward pass with $\gamma_{\text{inv}} = 1, \gamma_{\text{fwd}} = 7.5$ solves the problem, and the algorithm is able to generate the required 3D shape, however constant introduction of CFG on each step over-saturates the image. Using $\gamma_{\text{inv}} = -7.5, \gamma_{\text{fwd}} = 7.5$, however, is able to cancel the over-saturation bias and produce accurate 3D generations.

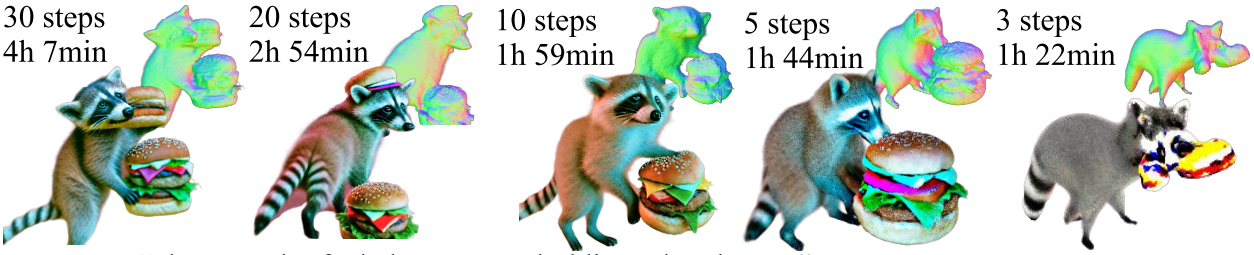

Figure 6.6: Ablation study of the proposed algorithm's performance based on the number of inversion steps.

Number of steps needed for DDIM inversion. On fig. 6.6 we ablate the number of steps needed for DDIM inversion. We conclude that $n = 10$ is the optimal choice that yields both good generation quality and does not slow down the generation too much.

Chapter 7

Conclusion, Limitations, and Future Work

In this work we have studied the discrepancy between high-quality image generation of 2D diffusion models and the blurry, often over-saturated results of Score Distillation Sampling (SDS) built on top of these diffusion models to generate 3D assets. We show that image guidance applied to each view of the 3D shape is in fact an approximation of a re-parametrized DDIM process. The only difference lies in the used noise: SDS randomly samples it from a Normal Gaussian i.i.d. on each iteration, while re-parametrized DDIM infers it conditionally on the previous steps. Random sampling of the noise breaks generative trajectories of DDIM and leads to over-smoothed results. We close the gap between the two algorithms and suggest to infer the noise via DDIM inversion. Through extensive ablation studies we show that DDIM inversion is an adequate approximation of the correct noise sample, and that simply adding it to SDS significantly improves visual quality of 3D generations. We compare our algorithm with other state-of-the-art 3D generations methods and prove that it achieves similar or better results, all while not requiring extra 3D supervision, training of separate diffusion models, or additional generation steps.

The final algorithm, however, has multiple limitations that we plan to address in the future. In this work we were focused on recovering the sampling quality of each view individually, but 3D consistency between these view still remains challenging. In particular, despite the proposed convexity loss, we still rarely observed that our algorithm can produce flat or concave "billboards". We envision that this problem can be solved with an external depth or normals supervision with the advances of the foundational image models. Another problem frequently reported across the field – content drift from one view to another of the generated 3D shape. Since there was no 3D supervision whatsoever, there is little to no communication between opposite views of the shape, which can lead to inconsistent 3D assets (e.g. a cat that has two tails on different sides). This problem might potentially be approached with stronger view conditioning, multi-view supervision, or the rising video generation models that combine huge, diverse datasets and multi-view data. Finally, score distillation is fundamentally capped by the performance of the underlying diffusion model and it is hence prone to similar hallucinations famously present in the original generative model (e.g. text and limbs anomalies). This also means that the algorithm inherits all the biases of the 2D diffusion model and can produce skewed distributions.

In conclusion, while our approach significantly improves the quality of 3D generation, further research is necessary to address these limitations. Future work could explore incorporating external depth or normals supervision, enhancing view consistency, and mitigating inherent biases and anomalies from the underlying diffusion models. By addressing these challenges, we aim to achieve high-quality reliable 3D asset generation.

Appendix A

Appendix

A.1 Suggested Algorithm

Algorithm 3 Our algorithm

Require:
$\psi \in \mathbb{R}^N$ - parametrized 3D shape
$\mathcal{C}$ - set of cameras around the 3D shape
y - text prompt
$g : \mathbb{R}^N \times \mathbb{C} \to \mathbb{R}^{n \times n}$ - differentiable renderer
$\epsilon_\theta^{(t)} : \mathbb{R}^{n \times n} \to \mathbb{R}^{n \times n}$ - trained diffusion model
Ensure: 3D shape ψ of y

procedure OURS(y)
  for i in range(n_iters) **do**
    $t \leftarrow 1 - i/n\_iters$ ▷ Use linear time anealing
    $c \leftarrow \text{Uniform}(\mathbb{C})$ ▷ Sample camera
    $\epsilon \leftarrow \kappa^{t+\tau}(x_0(t))$ ▷ Regress Noise via DDIM inversion
    $x_t \leftarrow \sqrt{\alpha(t)} g(\psi, c) + \sqrt{1 - \alpha(t)}\epsilon$ ▷ Noise current rendering
    $\nabla_\psi \mathcal{L}_{SDS} = \sigma(t) \left[\epsilon_\theta^{(t)}(x_t, y) - \epsilon \right] \frac{\partial g}{\partial \psi}$ ▷ Calculate the SDS update rule
    SGD update on ψ
  end for
end procedure

A.2 ODE derivation

In this section we show that Dreamfusion is an approximation of re-parametrized DDIM from the perspective of ODEs. To do the change of variable in eq. (3.1) to eq. (4.1) we need to differentiate it with respect to t and express $\frac{d\bar{x}(t)}{dt}$. Direct differentiation gives us:

$$\frac{dx_0(t)}{dt} = \frac{d\bar{x}(t)}{dt} - \epsilon_\theta^{(t)}\left(\frac{\bar{x}(t)}{\sqrt{\sigma(t)^2+1}}, y\right)\frac{d\sigma(t)}{dt} - \sigma(t)\frac{d}{dt}\epsilon_\theta^{(t)}\left(\frac{\bar{x}(t)}{\sqrt{\sigma(t)^2+1}}, y\right) \tag{A.1}$$

Now, expressing $\frac{d\bar{x}(t)}{dt}$ and merging with eq. (3.1) we get:

$$\epsilon_\theta^{(t)}\left(\frac{\bar{x}(t)}{\sqrt{\sigma(t)^2+1}}, y\right)\frac{d\sigma(t)}{dt} = \frac{dx_0(t)}{dt} + \epsilon_\theta^{(t)}\left(\frac{\bar{x}(t)}{\sqrt{\sigma(t)^2+1}}, y\right)\frac{d\sigma(t)}{dt} + \sigma(t)\frac{d}{dt}\epsilon_\theta^{t}\left(\frac{\bar{x}(t)}{\sqrt{\sigma(t)^2+1}}, y\right) \tag{A.2}$$

$$\begin{aligned}\frac{dx_0(t)}{dt} &= -\sigma(t)\frac{d}{dt}\epsilon_\theta^{(t)}\left(\frac{\bar{x}(t)}{\sqrt{\sigma(t)^2+1}}, y\right) \\ &= -\sigma(t)\frac{d}{dt}\epsilon_\theta^{(t)}\left(\frac{x_0(t)+\sigma(t)\kappa^t(x_0(t))}{\sqrt{\sigma(t)^2+1}}, y\right) \\ &= -\frac{d\kappa^t}{dt}(x_0(t))\end{aligned} \tag{A.3}$$

What gives us the DDIM's ODE re-parametrized for $x_0(t)$.

A.3 Out of distribution view.

Here we provide an intuition why it is so important to regress the noise κ instead of randomly sampling it. Lets look into the data that $\epsilon_\theta^(t)$ was trained on. For each particular t, the model has seen only clean images with the corresponding amount of noise $\sigma(t)$.

Let's consider a "bad" image $\hat{x}_0$. Out goal is to use our updated ODE to improve it and make it look more realistic. For t close to 1 the amount of noise is so big, that for any image $\hat{x}_0$ its noisy version $\hat{x}_0 + \sigma(t)\epsilon$ looks indistinguishable from the training data of the denoiser. **This explains why we can successfully use diffusion models to correct out-of-distribution images**.

For small t, however, the noise levels might be not enough to hide the artifacts of $\hat{x}_0$, so the image becomes out-of-distribution, and the predictions of the model become unreliable. We believe, that if instead of randomly sampling ϵ like in Dreamfusion, we can find such a noise sample, that under the same noise level t the artifacts of $\hat{x}_0$ will be hidden and it will be in-distribution of the model. This effectively increases the ranges of t where the predictions of the model are reliable and thus improves the generation quality. This intuition is illustrated in fig. A.1.

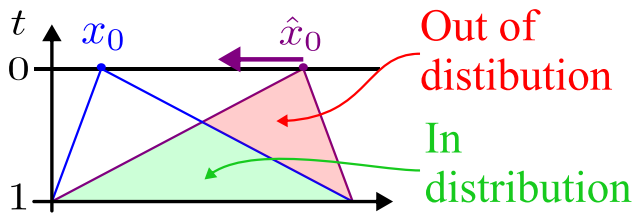

Figure A.1: For a training data point x_0 we consider how an out-of-distribution image $\hat{x}_0$ is perceived by the trained diffusion model. The cones depict how the image gets noised - notice that the cones meet at the bottom for the highest level of noise, however, for a big portion of timesteps t, $\hat{x}_0$ stays out of distribution for the diffusion model.

References

[1] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2022, pp. 10 684–10 695.

[2] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, *et al.*, "Laion-5b: An open large-scale dataset for training next generation image-text models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 25 278–25 294, 2022.

[3] W. Peebles and S. Xie, "Scalable diffusion models with transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4195–4205.

[4] A. C. Li, M. Prabhudesai, S. Duggal, E. Brown, and D. Pathak, "Your diffusion model is secretly a zero-shot classifier," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 2206–2217.

[5] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall, "Dreamfusion: Text-to-3d using 2d diffusion," *arXiv preprint arXiv:2209.14988*, 2022.

[6] H. Wang, X. Du, J. Li, R. A. Yeh, and G. 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*, 2023, pp. 12 619–12 629.

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

[8] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM transactions on graphics (TOG)*, vol. 41, no. 4, pp. 1–15, 2022.

[9] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics*, vol. 42, no. 4, pp. 1–14, 2023.

[10] J. Tang, J. Ren, H. Zhou, Z. Liu, and G. Zeng, "Dreamgaussian: Generative gaussian splatting for efficient 3d content creation," *arXiv preprint arXiv:2309.16653*, 2023.

[11] S. Earle, F. Kokkinos, Y. Nie, J. Togelius, and R. Raileanu, "Dreamcraft: Text-guided generation of functional 3d environments in minecraft," *arXiv preprint arXiv:2404.15538*, 2024.

[12] Z. Wang, C. Lu, Y. Wang, F. Bao, C. Li, H. Su, and J. Zhu, "Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[13] O. Katzir, O. Patashnik, D. Cohen-Or, and D. Lischinski, "Noise-free score distillation," *arXiv preprint arXiv:2310.17590*, 2023.

[14] J. Zhu, P. Zhuang, and S. Koyejo, "Hifa: High-fidelity text-to-3d generation with advanced diffusion guidance," in *The Twelfth International Conference on Learning Representations*, 2023.

[15] C.-H. Lin, J. Gao, L. Tang, T. Takikawa, X. Zeng, X. Huang, K. Kreis, S. Fidler, M.-Y. Liu, and T.-Y. Lin, "Magic3d: High-resolution text-to-3d content creation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 300–309.

[16] J. Tang, T. Wang, B. Zhang, T. Zhang, R. Yi, L. Ma, and D. Chen, "Make-it-3d: High-fidelity 3d creation from a single image with diffusion prior," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 22 819–22 829.

[17] Y. Chen, C. Zhang, X. Yang, Z. Cai, G. Yu, L. Yang, and G. Lin, "It3d: Improved text-to-3d generation with explicit view synthesis," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, 2024, pp. 1237–1244.

[18] S. Lin, B. Liu, J. Li, and X. Yang, "Common diffusion noise schedules and sample steps are flawed," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024, pp. 5404–5411.

[19] J. Ho and T. Salimans, "Classifier-free diffusion guidance," *arXiv preprint arXiv:2207.12598*, 2022.

[20] P. Wang, Z. Fan, D. Xu, D. Wang, S. Mohan, F. Iandola, R. Ranjan, Y. Li, Q. Liu, Z. Wang, *et al.*, "Steindreamer: Variance reduction for text-to-3d score distillation via stein identity," *arXiv preprint arXiv:2401.00604*, 2023.

[21] X. Yu, Y.-C. Guo, Y. Li, D. Liang, S.-H. Zhang, and X. Qi, "Text-to-3d with classifier score distillation," *arXiv preprint arXiv:2310.19415*, 2023.

[22] R. Liu, R. Wu, B. Van Hoorick, P. Tokmakov, S. Zakharov, and C. Vondrick, "Zero-1-to-3: Zero-shot one image to 3d object," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 9298–9309.

[23] Y. Shi, P. Wang, J. Ye, M. Long, K. Li, and X. Yang, "Mvdream: Multi-view diffusion for 3d generation," *arXiv preprint arXiv:2308.16512*, 2023.

[24] L. Melas-Kyriazi, I. Laina, C. Rupprecht, N. Neverova, A. Vedaldi, O. Gafni, and F. Kokkinos, "Im-3d: Iterative multiview diffusion and reconstruction for high-quality 3d generation," *arXiv preprint arXiv:2402.08682*, 2024.

[25] A. Blattmann, T. Dockhorn, S. Kulal, D. Mendelevitch, M. Kilian, D. Lorenz, Y. Levi, Z. English, V. Voleti, A. Letts, *et al.*, "Stable video diffusion: Scaling latent video diffusion models to large datasets," *arXiv preprint arXiv:2311.15127*, 2023.

[26] A. Tewari, T. Yin, G. Cazenavette, S. Rezchikov, J. Tenenbaum, F. Durand, B. Freeman, and V. Sitzmann, "Diffusion with forward models: Solving stochastic inverse problems without direct supervision," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[27] S. Szymanowicz, C. Rupprecht, and A. Vedaldi, "Viewset diffusion:(0-) image-conditioned 3d generative models from 2d data," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 8863–8873.

[28] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[29] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International conference on machine learning*, PMLR, 2015, pp. 2256–2265.

[30] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," *arXiv preprint arXiv:2011.13456*, 2020.

[31] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," *Advances in neural information processing systems*, vol. 34, pp. 8780–8794, 2021.

[32] M. Armandpour, A. Sadeghian, H. Zheng, A. Sadeghian, and M. Zhou, "Re-imagine the negative prompt algorithm: Transform 2d diffusion into 3d, alleviate janus problem and beyond," *arXiv preprint arXiv:2304.04968*, 2023.

[33] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *arXiv preprint arXiv:2010.02502*, 2020.

[34] P. Wang, D. Xu, Z. Fan, D. Wang, S. Mohan, F. Iandola, R. Ranjan, Y. Li, Q. Liu, Z. Wang, *et al.*, "Taming mode collapse in score distillation for text-to-3d generation," *arXiv preprint arXiv:2401.00909*, 2023.

[35] H. Hill and A. Johnston, "The hollow-face illusion: Object-specific knowledge, general assumptions or properties of the stimulus?" *Perception*, vol. 36, no. 2, pp. 199–223, 2007.

[36] Y.-C. Guo, Y.-T. Liu, R. Shao, *et al.*, *Threestudio: A unified framework for 3d content generation*, https://github.com/threestudio-project/threestudio, 2023.

[37] J. Hessel, A. Holtzman, M. Forbes, R. L. Bras, and Y. Choi, "Clipscore: A reference-free evaluation metric for image captioning," *arXiv preprint arXiv:2104.08718*, 2021.

[38] Nicki Skafte Detlefsen, Jiri Borovec, Justus Schock, Ananya Harsh, Teddy Koker, Luca Di Liello, Daniel Stancl, Changsheng Quan, Maxim Grechkin, and William Falcon, *TorchMetrics - Measuring Reproducibility in PyTorch*, Feb. 2022. DOI: 10.21105/joss.04101. URL: https://github.com/Lightning-AI/torchmetrics.

[39] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*, PMLR, 2021, pp. 8748–8763.

[40] J. Wang, K. C. Chan, and C. C. Loy, "Exploring clip for assessing the look and feel of images," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, 2023, pp. 2555–2563.

[41] R. L. Burden and J. D. Faires, "2.2 fixed-point iteration," *Numerical Analysis (3rd ed.). PWS Publishers*, p. 64, 1985.

[42] R. Mokady, A. Hertz, K. Aberman, Y. Pritch, and D. Cohen-Or, "Null-text inversion for editing real images using guided diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 6038–6047.

[43] D. Miyake, A. Iohara, Y. Saito, and T. Tanaka, "Negative-prompt inversion: Fast image inversion for editing with text-guided diffusion models," *arXiv preprint arXiv:2305.16807*, 2023.