DreamPropeller: Supercharge Text-to-3D Generation with Parallel Sampling

Supplementary Material

A. Theoretical Results

We show that the iteration rule in Eq. (9) satisfies the fixed-point property, and is guaranteed to converge to the true solution, i.e., the solution obtained by sequential computation which we denote as $\theta_0^\star, \ldots, \theta_T^\star$. Our result follows from a simple observation via induction. First, assume that after k fixed-point iterations, $\theta_\tau^k = \theta_\tau^\star$ for all $\tau \leq k$. We always initialize θ_0^k to be $\theta_0^\star$, so our inductive hypothesis trivially holds for $k = 0$. Next we examine the iteration rule for iteration $k+1$ and simplify terms using the pseudo-inverse property that $h^\dagger(s(\theta_\tau^\star), \theta_\tau^\star) = g(\theta_\tau^\star) = \theta_{\tau+1}^\star$.

$$
\begin{aligned}
\theta_{\tau+1}^{k+1} &= h^\dagger\left(s(\theta_\tau^k), \ldots h^\dagger\left(s(\theta_1^k), h^\dagger(s(\theta_0^k), \theta_0^k)\right)\right) \\
\theta_{\tau+1}^{k+1} &= h^\dagger\left(s(\theta_\tau^\star), \ldots h^\dagger\left(s(\theta_1^\star), h^\dagger(s(\theta_0^\star), \theta_0^\star)\right)\right) \\
\theta_{\tau+1}^{k+1} &= h^\dagger\left(s(\theta_\tau^\star), \ldots h^\dagger\left(s(\theta_1^\star), \theta_1^\star\right)\right) \\
\theta_{\tau+1}^{k+1} &= h^\dagger\left(s(\theta_\tau^\star), \ldots \theta_2^\star\right) \\
\theta_{\tau+1}^{k+1} &= h^\dagger\left(s(\theta_\tau^\star), \theta_\tau^\star\right) \\
\theta_{\tau+1}^{k+1} &= \theta_{\tau+1}^\star
\end{aligned}
$$

The above derivation shows that the inductive hypothesis extends to all $\tau \leq k+1$. Therefore, after at most T iterations the fixed-point iteration will converge to the true solution. In practice, however, the fixed-point iteration may converge with much fewer number of iterations.

B. Algorithm

A detailed algorithm is presented in Algorithm 2. We emphasize that since the computational units $s(\cdot)$ do not have nested dependencies, they can be computed *in parallel*. On the other hand, the $h^\dagger(\cdot)$ must be unrolled sequentially. Therefore, for speedup with parallel computation, $s(\cdot)$ should be chosen to contain most of the computational cost.

C. Additional Details on Practical Decisions

Sliding window. The sliding window scheme can be better understood with a simple example. For a window of size 3 containing $\{\theta_\tau^k, \theta_{\tau+1}^k, \theta_{\tau+2}^k\}$ with corresponding drifts $\{s(\theta_\tau^k), s(\theta_{\tau+1}^k), s(\theta_{\tau+2}^k)\}$, one slides the window forward by one if $\left\|\theta_{\tau+1}^{k+1} - \theta_{\tau+1}^k\right\|^2 \geq e$ and slides the window by two if $\left\|\theta_{\tau+1}^{k+1} - \theta_{\tau+1}^k\right\|^2 < e$ and $\left\|\theta_{\tau+2}^{k+1} - \theta_{\tau+2}^k\right\|^2 \geq e$.

We employ $\mathcal{L}_2$ norm on 3D parameter space in practice. In the case of generalized iteration, θ_τ^{k+1} and θ_τ^k may lie in different dimensions. We then calculate the $\mathcal{L}_2$ error between the two by $\left\|\theta_{\tau+1}^{k+1} - \text{unproj}(\theta_{\tau+1}^k)\right\|^2$. We denote this (squared) distance as $d(\theta_{\tau+1}^{k+1}, \theta_{\tau+1}^k)^2$ hereon. For the case of Adam optimizer, we choose to only measure the distance between θ parameters and ignore momentum parameters for convergence checking.

Algorithm 2 DreamPropeller

Input: 3D model with initial parameter θ, parameter dimension D, total time T, initial threshold e, error aggregation function M, Adam optimizer, EMA update γ, number of GPUs n.
Output: Score distillation output
$\tau, k, p \leftarrow 0, 0, n-1$
$\theta_i^0 \leftarrow \theta \quad \forall i \in [0, p-1]$
Initialize diffusion guidance (and optionally, LoRA for VSD) on p GPUs.
while $\tau < T$ **do**
  Dispatch $\{\theta_{\tau+j}^k\}_{j=0}^{p-1}$ to different GPUs and gather $\{s(\theta_{\tau+j}^k)\}_{j=0}^{p-1}$ with seed τ (Optionally, update LoRA separately on each GPU). // Gather $\nabla_\theta \mathcal{L}_{\text{SDS}}$ or $\nabla_\theta \mathcal{L}_{\text{VSD}}$
  $\theta_{\tau+j+1}^{k+1} \leftarrow h^\dagger(s(\theta_{\tau+j}^k), \ldots h^\dagger(s(\theta_\tau^k), \theta_\tau^k)\ldots), \quad \forall j \in [0, p-1]$
  error $\leftarrow \{\frac{1}{D} d(\theta_{\tau+j}^{k+1}, \theta_{\tau+j}^k)^2\}_{j=1}^p$
  skip $\leftarrow \min(\{j : \frac{1}{D} d(\theta_{\tau+j}^{k+1}, \theta_{\tau+j}^k)^2 > e, \quad \forall j \in [1, p]\} \cup \{p\})$
  $\theta_{\tau+j}^{k+1} \leftarrow \theta_{\tau+p}^{k+1} \quad \forall j \in [p, \text{skip}+p]$ // New window
  $\tau \leftarrow \tau + \text{skip}, \quad k \leftarrow k+1$
  $e \leftarrow \gamma e + (1-\gamma) * M(\text{error})$ // Adaptive threshold
  $p \leftarrow \min(p, T-\tau)$
end while
return θ_T^k

Eliminating stochasticity. We simply set the seed for time step τ to be τ. For increased variability, we can sample a random seed s before running the algorithm and set the seed for time step τ to be $\tau + s$.

Parallelizing Variational Score Distillation. We reuse all settings from Wang et al. [46] for each independent copies.

D. Experiments

Our code is based on `threestudio` [18][2] and for all the baselines we reuse their settings. Note that for TextMesh, `threestudio` only implements the coarse-stage generation. Therefore, as a wrapper around the package, we only experiment with coarse-stage TextMesh to demonstrate our

[2] https://github.com/threestudio-project/threestudio

	DreamFusion[29]	Magic3D [15]	TextMesh [44]	DreamGaussian [43]	ProlificDreamer [46]
Guidance Model	DeepFloyd	DeepFloyd+SD-2.1	DeepFloyd	SD-2.1	SD-2.1
Window Size p	7	7	7	7	7
Initial Threshold e ($\times 10^{-6}$)	5	5	5	500	50
Adaptivity γ	0.9	0.9	0.9	0.9	0.9
Error Aggregation Function M	median	median	median	median	mean
Batch Size	16	16	16	16	[8,2],8,1

Table 3. Default parameters used for Text-to-3D generation.

technique's effectiveness on this representation. For ProlificDreamer, for quantitative metrics, due to time limitations, we only report the runtime and quality metrics of the first stage (using NeRF). We observe similar speedup for its second/third stage during refinement. The qualitative comparisons, however, include the second and third stage refinement. The second stage also has batch size of 8 but for the third stage, we use batch size of 1 (since anything larger is prohibitively slow for baselines). Similarly, we build a lightweight wrapper for 3D Gaussian Splatting using the DreamGaussian implementation[3].

We summarize the default parameter settings relevant for our algorithm in Table 3 for Text-to-3D generation. All other parameters are held fixed from `threestudio` and `DreamGaussian` implementations. Note that for DreamFusion and Magic3D, we use DeepFloyd for coarse-stage generation and we use SD-2.1 for refinement.

For Image-to-3D generation, we also use the Zero-1-to-3 implementation from `threestudio`, which uses NeRF as the 3D representation, and `DreamGaussian`, which uses 3D Gaussian Splatting, respectively. For NeRF, we reuse settings from `threestudio`. For SDS, we progressively increase the rendering size in the order of $\{64, 128, 256\}$ and use batch size $\{16, 16, 10\}$ at step $\{0, 600, 900\}$. The single image size is also changed in the order of $\{128, 256, 512\}$. For 3D Gaussian Splatting, we use batch size 16 and reuse other settings from `DreamGaussian` implementation. We summarize relevant parameter settings in Table 4.

	NeRF [23]	3DGS [11]
Guidance Model	Zero-1-to-3	Zero-1-to-3
Window Size p	7	7
Initial Threshold e ($\times 10^{-6}$)	30	500
Adaptivity γ	0.9	0.9
Error Aggregation Function M	median	median
Batch Size	16,16,10	16

Table 4. Default parameters used for Image-to-3D generation using Zero-1-to-3 [17].

Evaluation settings. For Text-to-3D, we render a learned 3D shape from 12-degree elevation angle and 120 evenly-spaced azimuth angles all around. Each image is paired with the shape's input prompt. We use CLIP-B/32 for both R-Precision and FID calculation. The reference statistics for FID is the ImageNet 2012 validation set.

[3] https://github.com/dreamgaussian/dreamgaussian

E. Accelerating Image-to-3D Generation

Many works have also explored score distillation for Image-to-3D generation using 2D-diffusion finetuned on view-dependent data [16, 17, 30]. Among the most popular approaches is Zero-1-to-3 [17], which finetunes a large-scale diffusion model for novel-view synthesis given a single image and novel-view embeddings. It also serves as a powerful 3D-aware prior for score distillation, which luckily our framework can be directly applied to. In this section, we investigate our framework's application to the Image-to-3D generation task.

For evaluation, we choose NeRF [23] and 3D Gaussian Splatting [11] as the two representative examples for 3D representations with constant and changing dimensions during optimization. Each representation is equipped with Zero-1-to-3 and a source image for generating a novel 3D object, and we show that our framework can achieve substantial speedup when applied to Zero-1-to-3 while retaining generation quality. For both representations, we use batch size 16 unless otherwise noted, and we run SDS for 1200 steps and 500 steps respectively (details in Appendix D). We show 3 examples for each representation in Figure 6, where we compare novel views of the 3D results from Zero-1-to-3 and the results from Zero-1-to-3 accelerated by DreamPropeller.

We observe that our framework can achieve almost identical generation output with much shorter runtime for both representations. The wallclock time speedup is consistently more than 3x and 4x the original runtime. NeRF achieves lower speedup than the Text-to-3D counterpart due to the lower number of total steps compared to Text-to-3D generation (*e.g.* 25,000 steps), so the time for the costly initial model and data preloading for all GPUs is not effectively amortized. 3D Gaussian Splatting is less affected thanks to its lightweight representation conducive to fast cross-GPU communication and DreamGaussian's [43] simple implementation with minimal initial allocation cost.

F. Contribution and Acknowledgement

This research is done at Pika Labs and supported in part by NSF(#1651565), ARO (W911NF-21-1-0125), ONR (N00014-23-1-2159), CZ Biohub, HAI. Linqi Zhou led and worked on all parts of the project and Andy Shih proposed the initial generalized Picard iteration and contributed to framework formulation and initial experiments. We thank additional help and discussion with Chenlin Meng, Stefano Ermon, Felix Petersen, Wanqiao Xu and other colleagues at Stanford University.

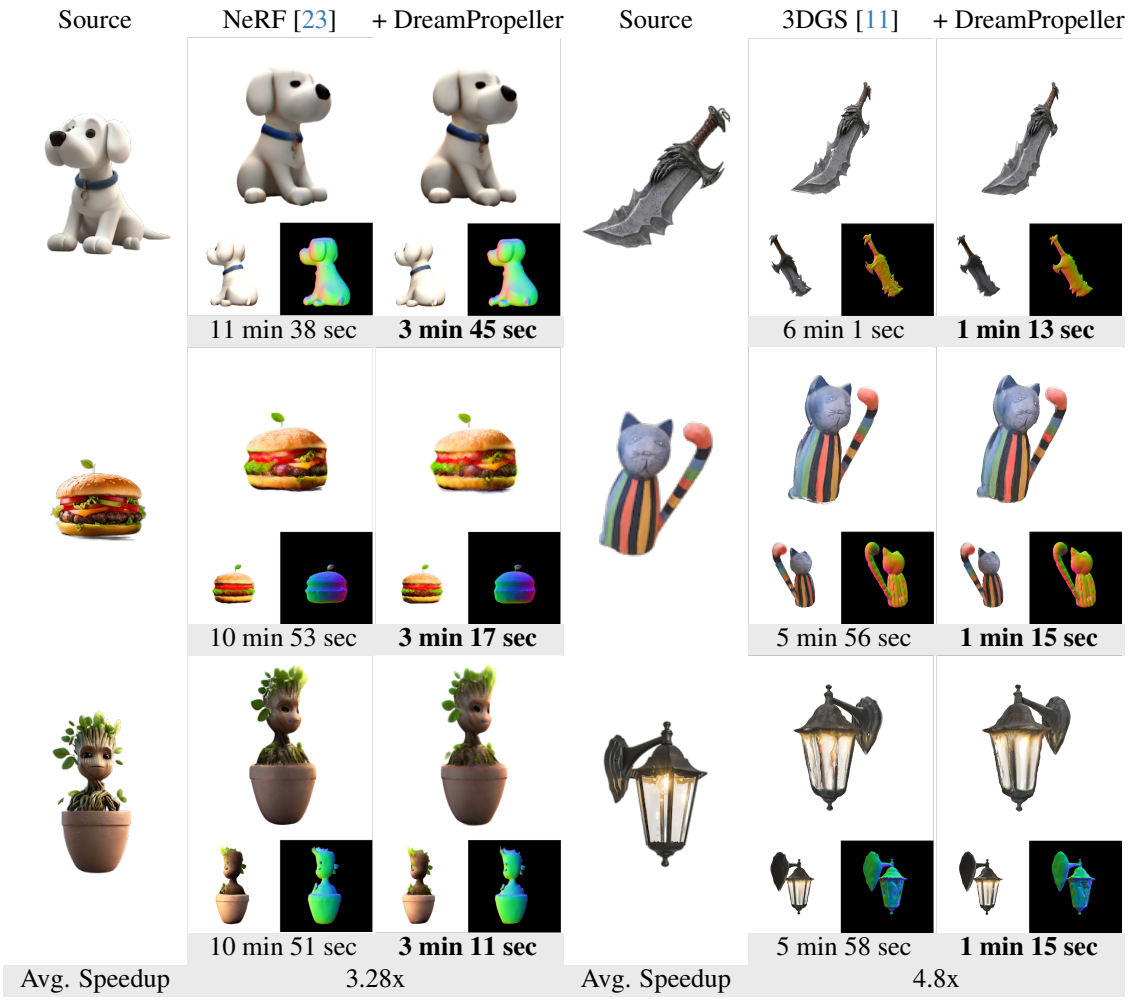

Figure 6. Visual comparisons for Image-to-3D generation using Zero-1-to-3 [17]. Our method achieves more than 3x and 4x average speedup respectively when applied to Zero-1-to-3 while achieving almost identical generation results.