DreamControl: Control-Based Text-to-3D Generation with 3D Self-Prior

Supplementary Material

In this supplementary, we provide more information on our implementation details (Section 1) and evaluation metrics (Section 2). To demonstrate the effectiveness of our control-based guidance on maintaining 3D prior, we provide a comparison with a two-stage ProlificDreamer (Section 3). For more visualization results in the format of 360°, please refer to the attached HTML file in *results* folder.

1. Implementation Details

Adaptive Viewpoint Sampling. We use a standard DreamFusion-IF framework in the first stage, in which the uniform viewpoint sampling is replaced with an adaptive sampling. To model an adaptive distribution, we evaluate the generation confidence in different views. The rotation angle ranges of the front view, the side view, and the back view are $[-60°, 60°)$, $[-120°, -60°) \cup [60°, 120°)$, $[-180°, -120°) \cup [120°, 180°)$, respectively. For each view, we denoise the latent code with a set of timesteps. Here, we set the timestep set T as $\{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$. We calculate the average CLIP similarity of all the timesteps, and softmax similarity scores of the three views as the view confidence. During optimization, the distribution of viewpoint sampling is proportional to confidence. Take generating corgi as an example, as shown in Figure 1, we denoise 3 views with 10 different timesteps. The generation confidences of front-view, side-view, and back-view are 68.75%, 4.75%, 26.51%. Consequently, the final probabilities of viewpoint distribution are $p_1^* = 68.75\%$, $p_2^* = 4.75\%$, and $p_3^* = 26.51\%$.

Boundary Integrity Metric. Our terminated condition in the first stage is related to the density difference between all valid rays and edge rays in NeRF scenes. The edge is detected by HEDdetector [6] implemented in ControlNet. We calculate the average density of each ray set and terminate the optimization when $\Delta_{\mathbf{r}}$ is less than 0.1 for three consecutive checkpoints. We set a checkpoint for every 100 iterations. We also provide an example of generating corgi in Figure 2. As the density difference decreases, the NeRF representation gradually forms a solid geometry.

Mesh Extraction. Our two-stage optimization can achieve promising visual results under the representation of NeRF, while the surfaces may exhibit severe irregularities if we export NeRF results to mesh objects for applications like animation. To obtain higher-quality 3D models for broader applications, we further refine the mesh representation based on DMTet [4]. We employ the same ControlNet guidance under NeRF representation to optimize the refinement of SDF representation. Specifically, we convert NeRF into a DMTet object and retain its texture information. We initially optimize its geometry to achieve smooth and regular surfaces. Subsequently, we optimized its texture to remove the noise generated during the geometry optimization. Note that, the optimization guidance in the SDF phase is exactly the same with the NeRF phase. The parameters of conditional LoRA on the NeRF training are loaded before the SDF optimization, which is frozen during the geometry phase and unfrozen during the texture phase.

Inference Time. In the the first stage, as the optimization is terminated automatically based on our metric, it usually takes less than 10 minutes for the generation of 3D self-prior. In the second stage, we optimize the generation for 15,000 iterations, which spend around 2 hours for a given prompt input. The total inference time is basically equivalent to previous text-to-3D methods like Fantasia3D [1] and ProlificDreamer [5]. For further extracting the mesh representation, since the color information from the NeRF phase is already of high quality, in the SDF phase, we only need to optimize the geometry and suppress texture noise appropriately. As a result, the entire process of mesh extraction is relatively fast, taking around 30 minutes.

Pseudocode. We provide pseudocode in Algorithm 1,2 to summarize the NeRF phase and the SDF phase in our framework, respectively.

2. Evaluation Metrics

Janus Rate. To evaluate the geometry consistency, we count the occurrence rate of the Janus problem (JR). As shown in Figure 3, we count generation as a Janus problem based on the following situations: (1) multi-face, multi-hand, multi-leg, or similar issues; (2) obvious content drift; (3) serious paper-thin generation. We calculate JR as the number of inconsistent content out of the number of all generated objects.

Pick-Score. Pick-Score (PS) [2] is a CLIP-based scoring model, which is trained with Pick-a-Pic, a large, open dataset of text-to-image prompts and real users' preferences over generated images. As a result, it can exhibit superhuman performance on the task of predicting human preferences. In 3D evaluation, we compare multi-view rendered images of generated objects to measure the preference. For a given text prompt y, we render N-view images $\{x_{y,i}^k\}_{i=1}^N$ from the generated object of method k. The pick score ps

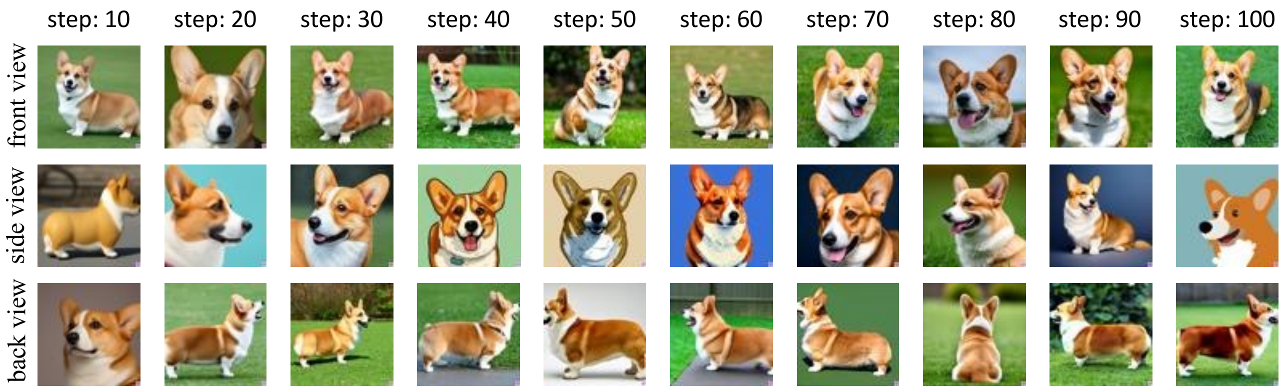

Figure 1. Visualization of generation in different view-dependent prompts. Generated corgis in the front view present a higher confidence than the other two views.

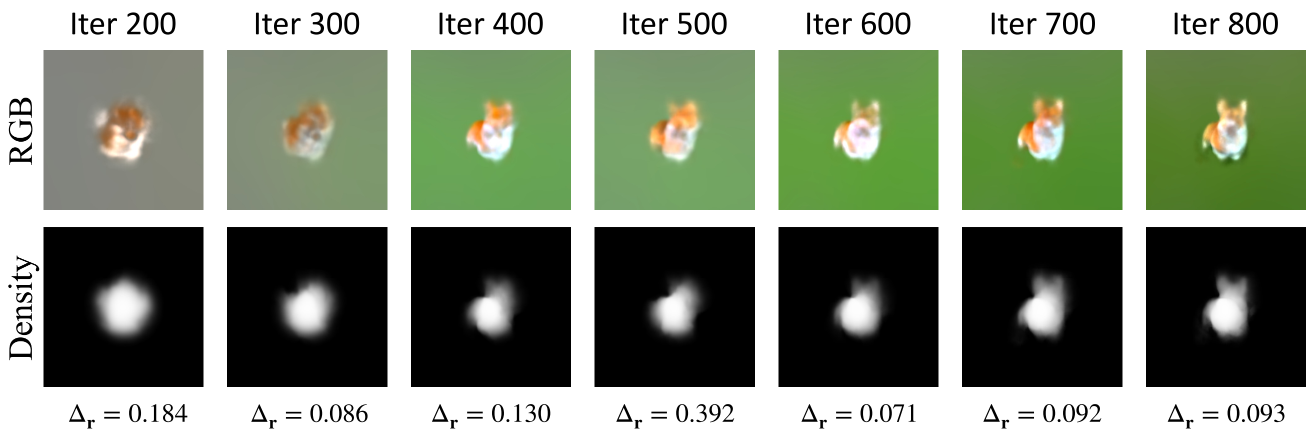

Figure 2. Visualization of the change in density difference Δ_r. As Δ_r decreases, the NeRF representation gradually forms a solid geometry.

for y is formulated as,

$$\text{ps}(y) = \frac{1}{N}\sum_{i=1}^{N} s_{\text{pick}}(y, [x^1_{y,i}, x^2_{y,i}, \dots, x^k_{y,i}]), \quad (1)$$

where s_{pick} is the CLIP-based scoring function. For each method k, the final pick score is calculated as the average score of all the text prompts, *i.e.*, $\frac{1}{|Y|}\sum_{y\in Y}\text{ps}(y,k)$.

CLIP-Score. CLIP-Score (CS) is based on the CLIP [3] similarity. For each prompt y, we render one image from the corresponding generated object of method k, x^k_y. The rendering viewpoint is a fixed camera pose at a 45° angle of elevation and a 30° angle of rotation. The final score is calculated as the average score of all the text prompts, *i.e.*, $\frac{1}{|Y|}\sum_{y\in Y} s_{\text{CLIP}}(y, x^k_y)$. Note that, differently from Pick-Score, we don't evaluate the CLIP-score on multi-view images because the Janus problem could ironically increase the CLIP similarity.

3. Two-Stage Comparison

To further demonstrate that our control-based guidance can maintain the 3D self-prior obtained in the first stage, we compare our second stage with two-stage ProlificDreamer [5]. For the two-stage ProlificDreamer, we use the 3D self-prior in our first stage for initialization.

As shown in Figure 4, we take the prompt "A chimpanzee dressed like Henry VIII king of England" as an example. The final generated chimpanzee of ProlificDreamer is quite different from its initialization, presenting multiple faces and multiple hands. Although DreamControl only uses the edge condition, our result can successfully maintain the geometry of the input condition. The results demonstrate that continually optimizing a coarse shape may still lead to overfitting issues.

Algorithm 1 DreamControl - NeRF Phase

```
1: **Input**: text prompt y
2: **Load**: stable diffusion v1.5 $\phi_{\text{sd}}$ and DeepFloyd XL-v1.0 $\phi_{\text{if}}$
3: ▶ **Preprocess: Viewpoint Confidence Analysis**
4: $y_{1,2,3} = y +$ ", front/side/back view"
5: **for** $t = 10, 20, \ldots, 100$ **do**
6:     Generate images $x^t_{y1}, x^t_{y2}, x^t_{y3}$ with $\phi_{\text{if}}$ in the timestep $t$
7: **end for**
8: $p^*_{1,2,3} = \text{softmax}(s_{\text{CLIP}}(y_{1,2,3}, [\{x^t_{y1}\}, \{x^t_{y2}\}, \{x^t_{y3}\}]))$
9: ▶ **Stage 1: 3D Self-Prior Generation**
10: **initialize** a NeRF scene $\theta$
11: **while** Not $\Delta_{\mathbf{r}} < 0.1$ for three consecutive checkpoints **do**
12:     Sample a camera $c$ based on $p^*$ and a timestep $t$
13:     Render $\theta$ at pose $c$, for RGB image $x = \mathbf{g}(\theta, c)$
14:     $\theta \leftarrow \theta - \mathbb{E}\left[\omega(t)\left(\hat{\epsilon}_{\phi_{\text{if}}}(x_t, t, y) - \epsilon\right)\frac{\partial x}{\partial \theta}\right]$
15: **end while**
16: 3D self-prior $\hat{\theta} = \theta$
17: ▶ **Stage 2: Control-Based Score Distillation**
18: **initialize** a NeRF scene $\theta$ and a conditional LoRA $\phi_\theta$
19: **while** not converged **do**
20:     Sample a camera $c$ and a timestep $t$
21:     Render $\theta$ for RGB image $x$ and normal map $x^n$ at pose $c$
22:     Render $\hat{\theta}$ and detect an edge mask $\hat{x}$ at pose $c$
23:     Update $\lambda$, weighted score $\mathcal{L}_{\text{score}}(x_t, \hat{x}_t, x^n_t, t) = (\hat{\epsilon}_{\phi_{\text{sd}}}(x_t, t, \hat{x}_t, y) - \epsilon) - \lambda(\hat{\epsilon}_\theta(x_t, t, x^n_t, y) - \epsilon)$
24:     $\theta \leftarrow \theta - \mathbb{E}\left[\omega(t)\mathcal{L}_{\text{score}}(x_t, \hat{x}_t, x^n_t, t)\frac{\partial x}{\partial \theta}\right]$
25:     $\phi_\theta \leftarrow \phi_\theta - \nabla_{\phi_\theta}\mathbb{E}\left[\|\hat{\epsilon}_\theta(\alpha_t x_t + \sigma_t\epsilon, t, c, y) - \epsilon\|_2^2\right]$ .
26: **end while**
27: **return** the optimized NeRF representation $\theta$
```

Algorithm 2 DreamControl - SDF Phase

```
1: **Input**: text prompt y
2: **Load**: $\phi_{\text{sd}}$, $\phi_\theta$, and $\theta$
3: ▶ **Stage 1: Geometry Refinement**
4: **initialize** a DMTet mesh from $\theta$, parameterized by $m$
5: **while** not converged **do**
6:     Sample a camera $c$ and a timestep $t$
7:     Render $m$ for RGB image $x$ and normal map $x^n$ at pose $c$
8:     Render $\hat{\theta}$ and detect an edge mask $\hat{x}$ at pose $c$
9:     $m \leftarrow m - \mathbb{E}\left[\omega(t)\left(\hat{\epsilon}_{\phi_{sd}}(x_t, t, y) - \hat{\epsilon}_\theta(x_t, t, x^n_t, y)\right)\frac{\partial x}{\partial m}\right]$
10: **end while**
11: ▶ **Stage 2: Texture Refinement**
12: **while** not converged **do**
13:     Sample a camera $c$ and a timestep $t$
14:     Render $m$ for RGB image $x$ and normal map $x^n$ at pose $c$
15:     Render $\hat{\theta}$ and detect an edge mask $\hat{x}$ at pose $c$
16:     $m \leftarrow m - \mathbb{E}\left[\omega(t)\left(\hat{\epsilon}_{\phi_{sd}}(x_t, t, y) - \hat{\epsilon}_\theta(x_t, t, x^n_t, y)\right)\frac{\partial x}{\partial m}\right]$
17:     $\phi_\theta \leftarrow \phi_\theta - \nabla_{\phi_\theta}\mathbb{E}\left[\|\hat{\epsilon}_\theta(\alpha_t x_t + \sigma_t\epsilon, t, c, y) - \epsilon\|_2^2\right]$ .
18: **end while**
19: **return** the optimized SDF representation $m$
```

4. More Visualization Results

Please refer to the attached HTML file in *results* folder for more visualization results in the format of 360° video.

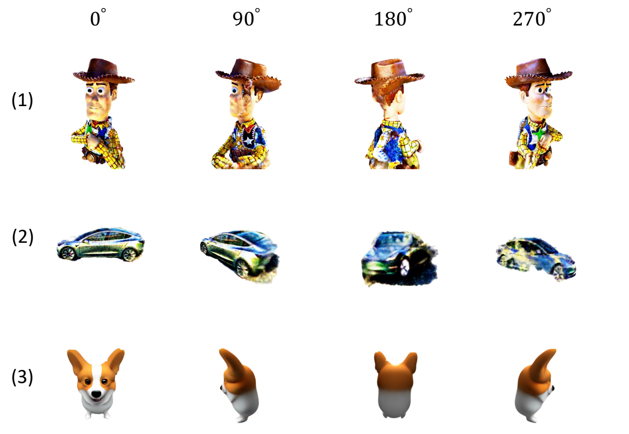

Figure 3. Visualization of Janus problems. (1) multi-face, multi-hand, multi-leg, or similar issues; (2) obvious content drift; (3) serious paper-thin generation.

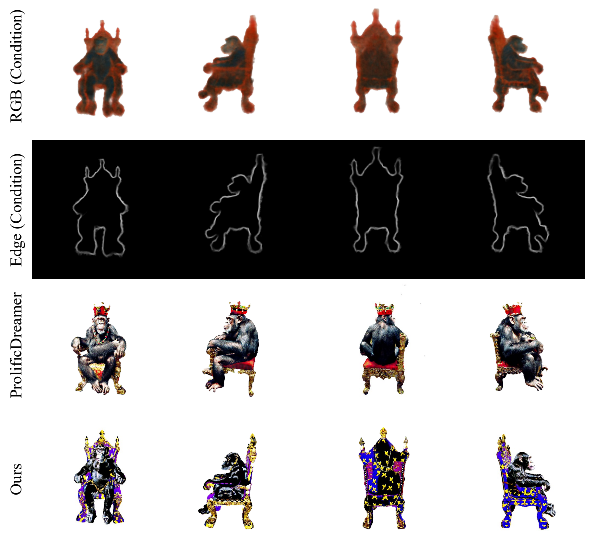

Figure 4. Visualization of two-stage generation comparison.

References

[1] 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)*, pages 22246–22256, 2023. 1

[2] Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-pic: An open dataset of user preferences for text-to-image generation. *arXiv preprint arXiv:2305.01569*, 2023. 1

[3] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learn-

ing transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 2

[4] 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. 1

[5] 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. *arXiv preprint arXiv:2305.16213*, 2023. 1, 2

[6] Saining "Xie and Zhuowen" Tu. Holistically-nested edge detection. In *Proceedings of IEEE International Conference on Computer Vision*, 2015. 1