

---

# TRIM: Scalable 3D Gaussian Diffusion Inference with Temporal and Spatial Trimming

## Supplementary Material

---

Anonymous Author(s)

Affiliation

Address

email

### 1 A Implementation Details

2 **Datasets.** For the text-to-3D generation task, T<sup>3</sup>Bench consists of three groups of prompts: a single  
3 object, a single object with surroundings and multiple objects. Each subgroup contains 100 text  
4 descriptions. For the image-to-3D generation task, due to the no access to the data selection list  
5 and rendering parameters used in baseline methods, we randomly select 300 objects from the GSO  
6 dataset. Each object is rendered from four orthogonal viewpoints: front, back, left, and right sides,  
7 with a fixed elevation angle of 0°.

8 **3D Generation Setting.** Our 3D generation model with the backbone of Stable-Diffusion-3.5-  
9 Medium uses the original flow matching Euler ODE solver with 28 steps in the main experiments. The  
10 classifier-free guidance scale is set to 7 for text-to-3D generation and 2 for image-to-3D generation.  
11 The image input is center cropped and resized to 256×256 resolution.

12 **Pairwise Data Synthesis.** We first use ChatGPT-4o to generate  $N = 100$  text prompts describing  
13 diverse objects with various decorations. Following Algorithm 1, we perform data synthesis by  
14 sampling  $M = 64$  denoising trajectories per prompt using distinct random seeds. For each trajectory,  
15 we decode and render the final denoised output  $\mathbf{z}_0$  and compute the alignment score  $s$  using a  
16 CLIP-based evaluator.

17 For data processing, we construct pairwise training and testing data from the 100 latent-score pairs.  
18 Each data point consisted of  $(\mathbf{z}_t^1, \mathbf{z}_t^2, s_1 - s_2)$ , where  $s_1 - s_2$  represents the difference between  
19 the scores of the two latent trajectories. The dataset was then split into a 7:3 ratio for training and  
20 testing, respectively. A smaller score distance indicates minimal differentiation between two latent  
21 trajectories, making accurate distinction challenging but less critical for selection. Conversely, a larger  
22 score distance signifies substantial differences, highlighting the importance of correct classification  
23 for effective selection. Due to the pairwise combination method, the data distribution is imbalanced,  
24 with a disproportionately large portion of data samples exhibiting small score distances. To ensure  
25 balanced data training, we group the data into 11 bins based on absolute score distance  $|s_1 - s_2|$ ,  
26 with intervals of 0.1, *i.e.*,  $[0, 0.1)$ ,  $[0.1, 0.2)$ , ...,  $[1, \infty)$ . We ensure that each group contains 200  
27 samples, resulting in a total training dataset of 2200 samples.

28 **Selector Training Setting.** We employ the AdamW optimizer with a learning rate of 0.001, a  
29 weight decay of 0.01, and a cosine weight decay schedule. Training is conducted with a batch size  
30 of 64 for 20 epochs. The model takes a pair of latent features  $(\mathbf{z}_t^1, \mathbf{z}_t^2)$  as input and predicts the  
31 pairwise comparison outcome. The target label is defined as  $\mathbb{1}(s_1 > s_2)$ , and the loss is computed  
32 using binary cross-entropy over the softmaxed prediction. The trained selector achieves over 70%  
33 test accuracy on the overall pairwise validation set in Table 3 and exceeds 90% accuracy for test  
34 samples with large score gaps ( $|s_1 - s_2| > 1$ ). This indicates that the well-trained selector effectively  
35 captures discriminative features in the latent space and is capable of reliably identifying higher-quality  
36 trajectories during inference.

---

**Algorithm 1** Data Synthesis for Latent Selector Training

---

```
1: Input: Prompt set  $\mathcal{P} = \{p_i\}_{i=0}^N$ , # trajectories per prompt  $M$ , # denoising steps  $T$ , camera  
   matrix  $\mathbf{C}_M$ , decoder, renderer, and evaluator models.  
2: Output: A triplet dataset  $\mathcal{D} = \{(p_j, \{(\mathcal{T}_i, s_i)\}_{i=1}^M)\}_{j=1}^N$   
3: Initialize empty dataset  $\mathcal{D} \leftarrow \emptyset$   
4: for  $j = 1$  to  $N$  do ▷ Iterate over prompts  
5:   Initialize empty set  $\mathcal{S}_j \leftarrow \emptyset$  ▷ Store trajectory-score pairs for prompt  $p_j$   
6:   for  $i = 1$  to  $M$  do ▷ Iterate over random seeds  
7:     Set random seed  $r_i \leftarrow i$   
8:     Generate trajectory  $\mathcal{T}_i = \{\mathbf{z}_T^i, \mathbf{z}_{T-1}^i, \dots, \mathbf{z}_0^i\} \leftarrow \mathcal{G}(p_j, r_i)$   
9:     Decode final latent  $\mathcal{O}_i \leftarrow \text{Decoder}(\mathbf{z}_0^i)$   
10:    Render images  $\mathcal{I}_i \leftarrow \text{Renderer}(\mathcal{O}_i, \mathbf{C}_M)$   
11:    Evaluate quality  $s_i \leftarrow \text{Evaluator}(\mathcal{I}_i)$   
12:    Add pair to set  $\mathcal{S}_j \leftarrow \mathcal{S}_j \cup \{(\mathcal{T}_i, s_i)\}$   
13:   end for  
14:   Add prompt and pairs to dataset  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(p_j, \mathcal{S}_j)\}$   
15: end for  
16: return  $\mathcal{D}$ 
```

---

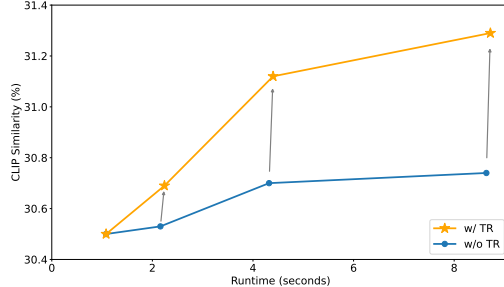
## B Experiment results

**Computation Cost.** Table 1 presents the ablation study on the computational contributions of the proposed Trajectory Reduction (TR) and Instance Masking (IM) components. We observe that TR significantly reduces the overall FLOPs by decreasing the total number of denoising steps. However, since multiple trajectories are processed in parallel and the runtime is determined by the longest trajectory, the actual GPU memory usage and runtime show a slight increase due to the additional cost introduced by the selector model. In contrast, IM reduces the computational load of the denoising transformer by pruning background tokens, resulting in clear improvements in both throughput and runtime. Thus, the combined TRIM balances temporal and spatial efficiency, leading to improved overall inference performance across all computational cost metrics.

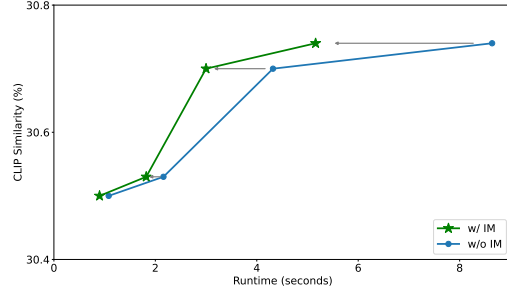
**Table 1:** Computation resource comparison.

	FLOPs (T)↓	Mem. (GB)↓	Throughput (step/s)↑	Runtime (second)↓
SD-3.5-Medium	195.68	33.26	13.18	8.64
+ IM	165.60	32.85	18.09	5.16
+ TR	110.07	33.55	13.18	8.74
+ TRIM	106.31	33.13	18.09	5.24

**Ablation on TRIM.** We analyze the contributions of Trajectory Reduction (TR) and Instance Masking (IM) in Figure 1 and Figure 2, respectively. We observe that TR primarily improves the CLIP similarity score of the generated 3D outputs with a slight increase in processing time, while IM is more effective at reducing overall runtime while maintaining similar CLIP scores. Thus, as shown in Figure 7 of the main paper, combining both components in TRIM leads to improvements in both generation quality and inference efficiency.



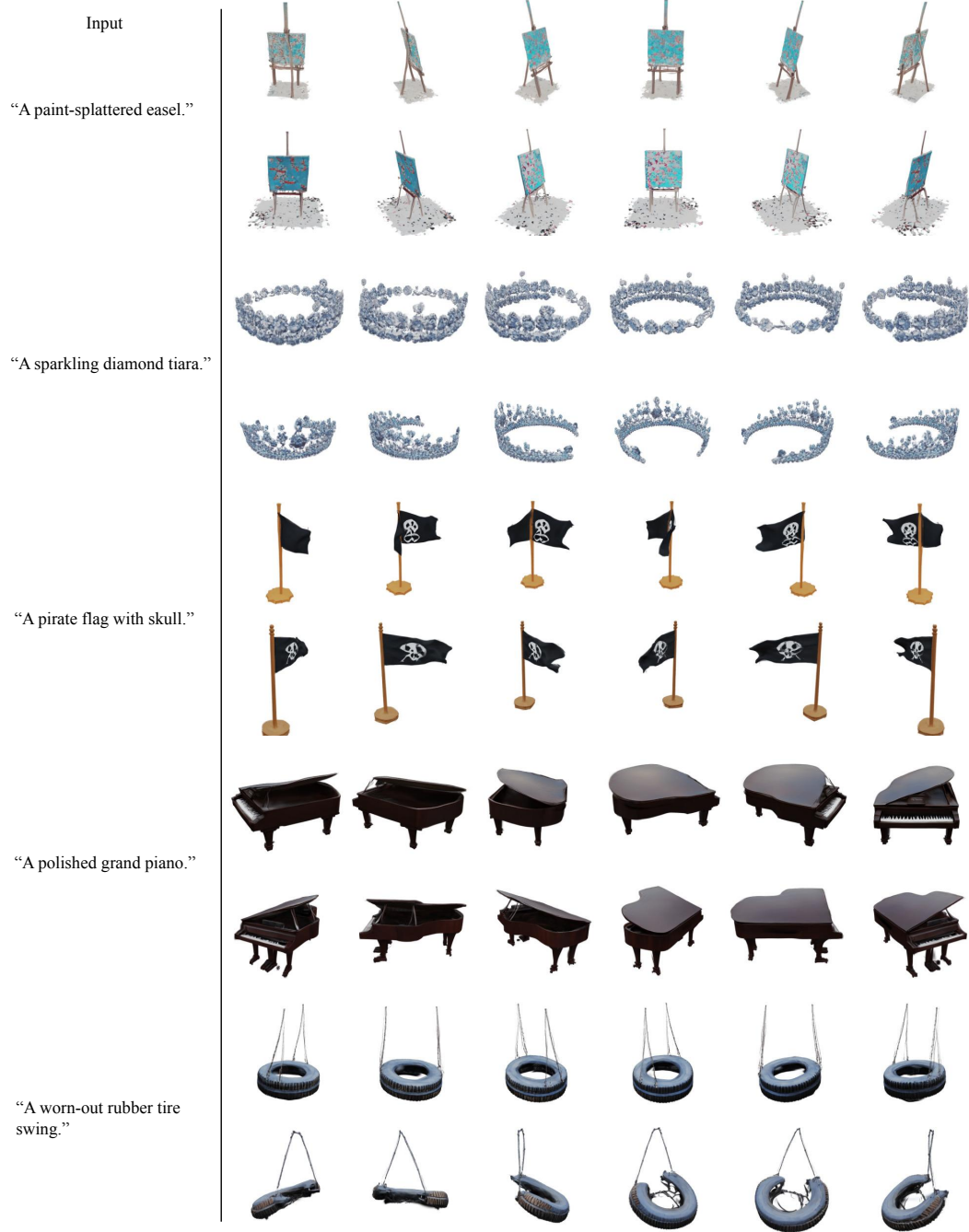
**Figure 1:** Ablation results on TR.



**Figure 2:** Ablation results on IM.

## 53 **C More Visualization**

54 We provide more visualization comparisons on text/image-to-3D generation from DiffSplat and TRIM  
55 in this section. Figure 3 shows that TRIM captures finer details, such as the “splattered” colors around  
56 the easel and the “worn-out” characteristics of the tire, demonstrating better alignment with the input  
57 text prompts compared to DiffSplat. Figure 4 demonstrates that TRIM produces more structurally  
58 consistent 3D shapes with fewer distortions, leading to more realistic and coherent generation.



**Figure 3:** Visualization comparisons on T<sup>3</sup>Bench from DiffSplat (Rows 1,3,5,7&9) and TRIM (Rows 2,4,6,8&10).



**Figure 4:** Visualization comparisons on GSO dataset from DiffSplat (Rows 1,3,5,7&9) and TRIM (Rows 2,4,6,8&10).