

GSV3D: Gaussian Splatting-based Geometric Distillation with Stable Video Diffusion for Single-Image 3D Object Generation

Supplementary Material

Our supplementary material gives more details about our method and more experimental results, which can be summarized as follows.

- We provide the detailed dataset preparation procedure in Section 1.
- We provide the network details in Section 2.
- We provide details of the user study in Section 3.
- We provide the performance metric of our method in Section 4.
- We provide the details of mesh extraction in Section 5.
- We provide visualization of the geometric distillation ablation study in Section 6.
- We provide more quantitative comparisons in Section 7.
- We provide more qualitative comparisons in Section 8.

1. Dataset Preparation

To construct a training dataset, we collect a set of 3D models from the Objaverse dataset. However, the Objaverse [2] dataset contains a mix of data of varying quality, including low-quality 3D scans and sub-par geometric models, which could negatively impact training. To address this, we manually remove damaged and non-watertight 3D models. Subsequently, we render these remaining 3D models in Blender. During rendering, we normalize the models to fit within a unit sphere. We set the camera’s field of view (FOV) to 50 degrees and the rendering resolution to 576×576 . The camera is positioned at a distance of 2.7 units from the object, and for each object, we randomly select an elevation angle between -5 and 30 degrees. The camera is rotated 360 degrees around the object, beginning at 0 degrees azimuth, to uniformly render 84 RGB images and depth maps. Notably, to avoid harsh shadows caused by point lighting, we use HDRI environment lighting. With this setup, we ultimately render approximately 840,000 images in total.

2. Network Details

2.1. Multi-view Diffusion Details

In this section, we provide a detailed description of the multi-view diffusion model mentioned in the submitted paper. Our method is built upon SV3D, which shares a similar architecture with Stable Video Diffusion [1]. We initialize the model parameters from SV3D-P and follow all hyperparameter settings of SV3D. Then we froze the whole model and injected LoRA (Low-Rank Adaptation) into the UNet, setting the LoRA parameters to be trainable to maintain the ability of the pre-trained model as much as possible. We

use the cloneofsimo version of LoRA [3], configured with a rank of 16 and no bias. When the input resolution is set to 576×576 , the number of frames N , the height H , the width W , and the channel C of the noisy multi-view latents $z \in \mathbb{R}^{N \times H \times W \times C}$ are 16, 72, 72, and 4. The attention mechanism uses 5 heads. The training process is conducted with 4 NVIDIA 80G-A100GPUs. We train our model with a constant learning rate of $5e^{-5}$ with a batch size of 2. During inference, we adopt the default sampling schedule of SV3D with 50 sampling steps. The scale of classifier-free guidance (CFG) is set to 2.5 in our experiments.

2.2. Gaussian Splatting Decoder Details

We provide a detailed description of the architecture of the Gaussian Splatting Decoder network, which is designed to extract implicit geometric features from the latent space and convert them into an explicit 3D representation through Gaussian Splatting. With this decoder, we can not only convert latents to Gaussian Splatting directly but also distill the multi-view diffusion model to obtain much more consistent results. At the core of this decoder is a transformer encoder, or ViT as shown in Figure 3 in the submitted paper, which is employed to model the relationships between the input multi-view latents \hat{z}_t which contains $N = 16$ frames. We begin by concatenating the input multi-view latents (with an original channel dimension C of 4) with camera positional embeddings generated by a Plucker embedder, which increases the channel dimension from C to $(C + 6)$. We then patchify the multi-view latents into tokens through a two-step process. We tokenize the multi-view latents using 2D convolutional network with a kernel size of 3 and a stride of 2. This reduces the spatial dimensions from $N \times H \times W \times (C + 6)$ to $N \times H//2 \times W//2 \times 768$, where the height H is 72, the width W is 72, and the channel dimension is transformed from $(C + 6)$ to 768. Subsequently, a 3D convolutional network with a kernel size of 3 and a stride of 4 is employed to merge these N views tokens into $N//4$ views, resulting in a feature \mathbf{F}_{ViT} with a shape of $N//4 \times H//2 \times W//2 \times 768$. Additionally, we leverage the powerful pre-trained model DINOv2 [4] to extract semantic information from the conditioning image \mathbf{R} . Since DINOv2 employs a 14×14 patch strategy, where the input image is divided into a grid of 14 patches along both height and width, the conditioning image must be resized from 576×576 to 504×504 to ensure the final output feature \mathbf{F}_{DINO} has a shape of $36 \times 36 \times 768$. By incorporating multi-head cross attention in the transformer

encoder, the Gaussian Splatting decoder utilizes this semantic information to complement the features that the multi-view latents may lack. In each layer of the transformer encoder, cross-attention is first applied between the features \mathbf{F}_{ViT} and the features \mathbf{F}_{DINO} , where the features \mathbf{F}_{ViT} serve as the query (Q), and the features \mathbf{F}_{DINO} act as the key (K) and value (V). This cross-attention operation is performed along the sequence length dimension, enabling the model to effectively fuse semantic information from DINO with the multi-view latents. Following the cross-attention, the output is passed to a self-attention layer, where the query, key, and value are all derived from the same input. Both the cross-attention and self-attention mechanisms use 16 attention heads. Subsequently, the output of the self-attention is fed into a multi-layer perceptron (MLP). The MLP consists of a linear layer that expands the channel dimension by a factor of 4, followed by a GeLU activation function, and finally, another linear layer that reduces the dimension back to its original size. Finally, the output of the ViT is passed to an upsampler, which comprises multiple layers of PixelShuffle and attention mechanisms. A linear layer serves as the final layer to predict the parameters of Gaussian Splatting.

3. Details of User Study

Figure 1 presents an example of our user study. In the user study, volunteers are provided with seven videos rendered from generated 3D representations, one of which is generated using our method, while the remaining six are produced by comparative methods. The volunteers are asked to complete the following tasks: 1) Please rank these results according to their geometry quality. (Without ghosting or distortions) 2) Please rank these results according to their appearance quality. (Most similar to the input image)

4. Performance Metrics

This section provides a detailed analysis of the model’s inference time, the number of parameters, and GPU memory usage when processing an input image at a resolution of 576×576 on an NVIDIA A100-80G. The inference time is 48.45 seconds with the inference step set to 50. The total number of parameters in the model is 2.7 billion, with the majority concentrated in the convolutional network and attention mechanisms. For GPU memory usage, during inference, the peak memory consumption reaches 15 GB. These metrics provide critical insights for model deployment and optimization.

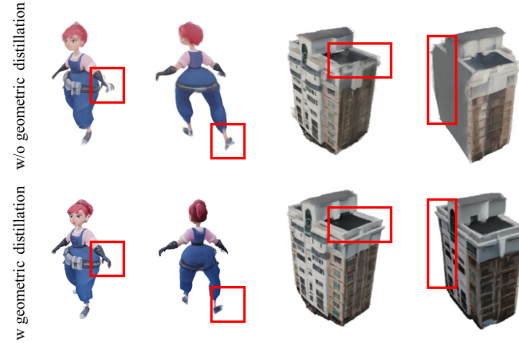
5. Details of Mesh extraction

To obtain Mesh from Gaussian Splatting for geometry evaluation, we randomly sample over 200 camera views on a sphere and render both RGB images and depth maps from

these viewpoints to obtain an explicit 3D mesh. We perform truncated signed distance function (TSDF) fusion using Open3D [5] to integrate the reconstructed depth maps. During TSDF fusion, we set the voxel size to 0.004 and the truncation threshold to 0.02.

6. Visualization of Geometric Distillation Ablation Study

More visualization of the geometric distillation ablation study has been shown below:



7. More Quantitative Comparisons

We evaluate Wonder3D, GRM, and InstantMesh on the GSO dataset and compare all the baselines on Objaverse datasets and real world cases in OmniObject3D dataset using the same experimental setup as mentioned in the submitted paper. Below are the updated quantitative results:

Method	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	FID \downarrow	KID \downarrow	CLIP-IQA \uparrow	CD \downarrow	IoU \uparrow	F-Score \uparrow
Wonder3D	18.324	0.862	0.035	87.26	0.86	0.817	0.213	0.467	0.452
GRM	17.742	0.861	0.033	73.57	0.79	0.810	0.198	0.415	0.305
InstantMesh	16.285	0.848	0.034	67.51	0.54	0.805	0.248	0.392	0.268
GSV3D	20.390	0.884	0.023	50.79	0.21	0.839	0.100	0.721	0.661

Table 1. Quantitative comparisons on GSO dataset.

Method	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	FID \downarrow	KID \downarrow	CLIP-IQA \uparrow	CD \downarrow	IoU \uparrow	F-Score \uparrow
GA	16.129	0.868	0.038	89.45	1.10	0.809	0.197	0.515	0.318
TGS	19.948	0.879	0.031	83.91	1.18	0.830	0.270	0.426	0.237
LGM	18.651	0.867	0.033	56.10	0.43	0.858	0.237	0.450	0.321
Zero123Plus	16.477	0.847	0.037	79.30	0.77	0.828	0.164	0.535	0.418
Era3D	15.785	0.843	0.040	88.01	0.85	0.825	0.228	0.447	0.257
SV3D	18.527	0.871	0.034	70.84	0.80	0.833	0.198	0.438	0.312
Wonder3D	19.507	0.861	0.033	85.63	0.78	0.820	0.204	0.472	0.468
GRM	18.499	0.867	0.034	72.70	0.77	0.845	0.191	0.418	0.315
InstantMesh	16.361	0.856	0.033	60.94	0.51	0.832	0.242	0.398	0.269
GSV3D	22.256	0.897	0.022	49.27	0.19	0.876	0.098	0.756	0.690

Table 2. Quantitative comparisons on Objaverse dataset.

Method	PSNR \uparrow	SSIM \uparrow	LPIPS \downarrow	FID \downarrow	KID \downarrow	CLIP-IQA \uparrow	CD \downarrow	IoU \uparrow	F-Score \uparrow
GA	13.941	0.809	0.040	109.67	1.23	0.770	0.201	0.497	0.303
TGS	18.501	0.851	0.033	97.06	1.40	0.784	0.272	0.404	0.223
LGM	16.869	0.840	0.034	64.94	0.47	0.799	0.247	0.423	0.299
Zero123Plus	14.236	0.826	0.037	92.34	0.90	0.796	0.172	0.513	0.403
Era3D	14.034	0.798	0.042	95.81	0.93	0.804	0.232	0.430	0.243
SV3D	16.620	0.840	0.034	86.04	0.80	0.811	0.209	0.415	0.307
Wonder3D	18.093	0.851	0.036	98.40	0.92	0.785	0.216	0.462	0.439
GRM	16.301	0.834	0.033	84.59	0.78	0.810	0.205	0.396	0.291
InstantMesh	16.164	0.822	0.036	72.77	0.59	0.805	0.257	0.390	0.258
GSV3D	19.209	0.873	0.025	58.17	0.23	0.818	0.104	0.687	0.647

Table 3. Quantitative comparisons on OmniObject3D dataset.

Notably, the metrics of Wonder3D differs from the original paper mainly due to different GSO objects we sampled and different elevation angles we set during rendering.

8. More Qualitative Comparisons

Figure 2 and Figure 3 provide more qualitative comparisons on the GSO dataset and some real-world cases. Our method is capable of generating results with better geometry consistency than the compared methods while simultaneously being realistic. Each case includes two distinct rendering views of the generated 3D representation. For dynamic visualization, please refer to the supplementary video provided with this material.

References

- [1] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023. [1](#)
- [2] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *CVPR*, pages 13142–13153, 2023. [1](#)
- [3] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. [1](#)
- [4] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision, 2023. [1](#)
- [5] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing. *arXiv preprint arXiv:1801.09847*, 2018. [2](#)

3d generation user study

Here is an input image. We aim to generate a 3D model according to this image.



The following 7 videos are rendered from generated 3D representations of different methods.



* 1. Please rank these results according to their geometry quality. (Without ghosting or distortions)

- ☐ 1st video
- ☐ 2nd video
- ☐ 3rd video
- ☐ 4th video
- ☐ 5th video
- ☐ 6th video
- ☐ 7th video

* 2. Please rank these results according to their appearance quality. (Most similar to the input image)

- ☐ 1st video
- ☐ 2nd video
- ☐ 3rd video
- ☐ 4th video
- ☐ 5th video
- ☐ 6th video
- ☐ 7th video

Figure 1. One comparison example in the user study. In each comparison, the volunteers are asked the following task: 1) Please rank these results according to their geometry quality. (Without ghosting or distortions) 2) Please rank these results according to their appearance quality. (Most similar to the input image)

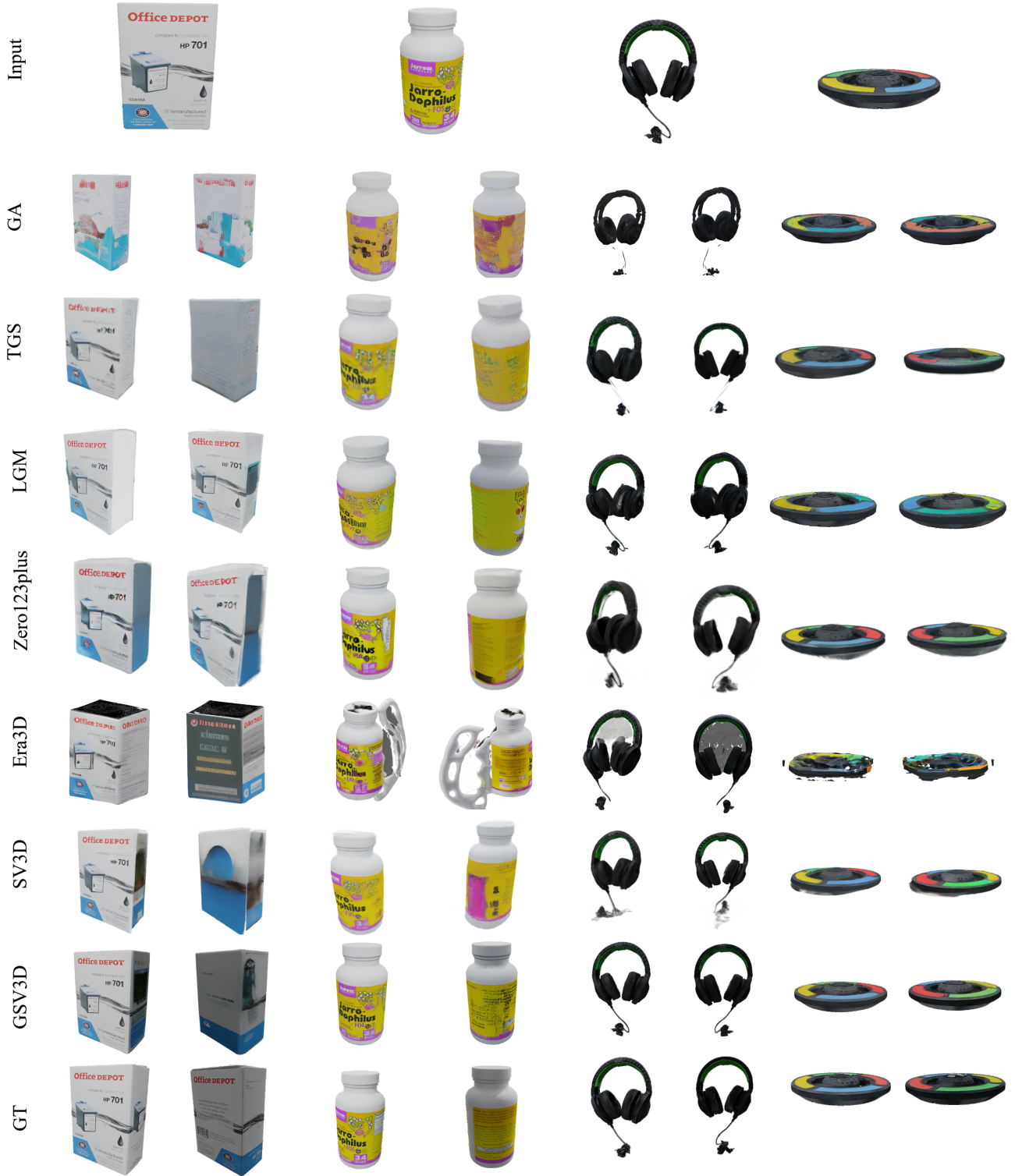


Figure 2. Performance comparison between our GSV3D and other state-of-art methods on GSO dataset. Each case provides two distinct rendering views of the generated 3D representations. GA and TGS are abbreviations for GaussianAnything and TriplaneGaussian, respectively.

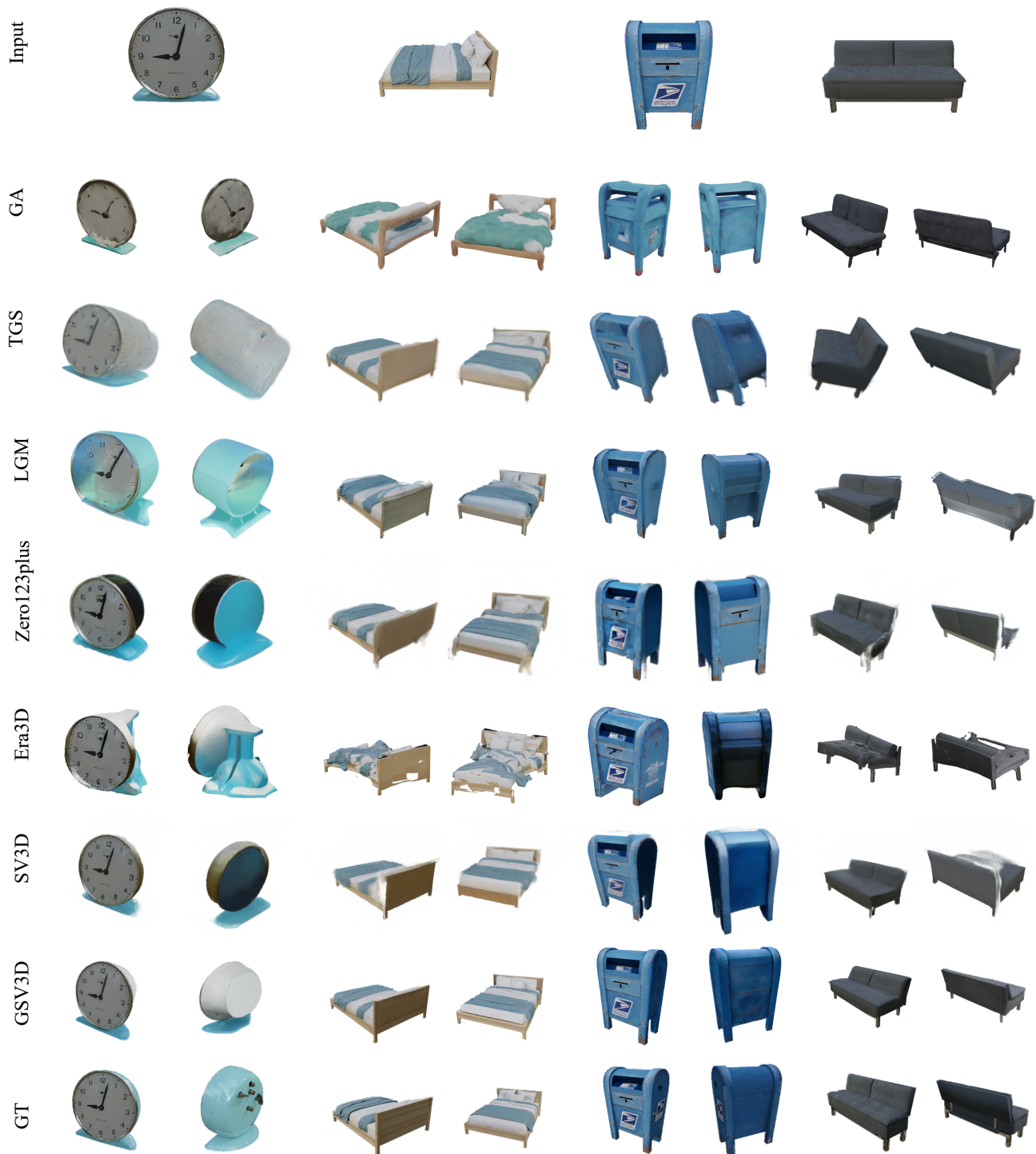


Figure 3. Performance comparison between our GSV3D and other state-of-art methods on real-world cases. Each case provides two distinct rendering views of the generated 3D representations. GA and TGS are abbreviations for GaussianAnything and TriplaneGaussian, respectively.