3D-Fixer: Coarse-to-Fine In-place Completion for 3D Scenes from a Single Image

Supplementary Material

1. Implementation details

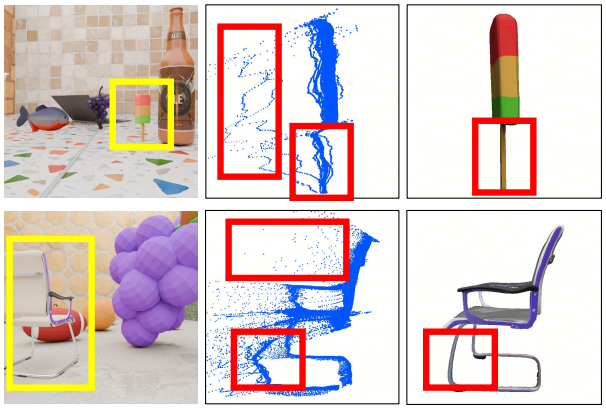
Base model. Our method builds on the TRELLIS [8] framework, which is a two-stage rectified flow generation method, where the first stage is a DiT [5] model that generates the sparse voxel structure in the latent space, and the second stage is a DiT-style model based on the sparse coordinates predicted in stage one to generate the SLAT representation [8]. The first stage utilizes 3D VAE to compress the 64^3 volumetric grid with binary occupancy into a low-resolution feature grid at a resolution of 16 and each grid contains a vector of dimension 8. The SLAT representation in TRELLIS is sparse volumetric feature representation, which is a sparse voxel grid at a resolution of 64 and each activated voxel stores an 8-dimensional feature vector. To construct this representation, TRELLIS first voxelizes a 3D asset into a sparse grid at resolution 64^3. It then renders dense views around the 3D asset and extracts per-view features via DINOv2 model [4]. These image features are subsequently projected onto the corresponding sparse voxels. Finally, a sparse 3D VAE encodes the aggregated DINOv2 features within each voxel into a compact 8-dimensional latent feature, producing the final SLAT representation.

3D-Fixer details. Our complete 3D-Fixer framework consists of three modules: the Coarse Structure Completer, the Fine Shape Refiner, and the Occlusion-Aware 3D Texturer. The three components are constructed based on image-conditioned TRELLIS model.

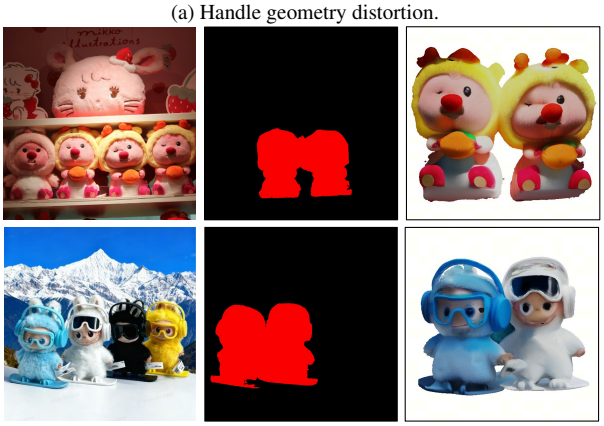
The Coarse Structure Completer and the Fine Shape Refiner are designed to generate the voxel grids, which are built on the first stage of TRELLIS model and each consists of 12 layers of our basic block as in Fig. 3 of the main paper. During training, we randomly sample an estimated depth map d_{est} from one of MoGe v2 [7], VGGT [6], DepthAnything v2 [9], or Depth Pro [1]. The sampled depth map is mixed with the ground truth depth map d_{gt} using a coefficient α as $d = \alpha \cdot d_{\text{est}} + (1 - \alpha) \cdot d_{\text{gt}}$, where α is uniformly sampled in $[0.0, 1.0]$. The α is further encoded as a depth-ratio embedding and provided to the model; during inference, we set α to 1.0. The visible point cloud is voxelized into a 64^3 volumetric grid, encoded into the latent space via the pre-trained 3D VAE from TRELLIS, and then supplied to both the Coarse Structure Completer and the Fine Shape Refiner as the partial geometry features. For the global geometry conditioning, we use the MoGe v2 [7] to extrach feature tokens from the scene image, while the occluded conditioning is provided by instance-level image tokens from DINOv2.

The Occlusion-Aware 3D Texturer generates textured 3D assets conditioned on the voxels produced by the first stage, which is built on the second-stage TRELLIS architecture and similarly comprises 12 layers of our basic block. To provide 3D-aware texture cues, we project the scene-level global features onto the voxel grid. Additionally, we calculate the visibility ratio of the voxel grid with respect to the input view and encode this value as a visibility ratio embedding, which is also supplied to the model. The global geometry conditioning and occlusion-aware conditioning follow the same design as in our first-stage models.

Training. We train the 3D-Fixer on our proposed dataset. Because 3D instances in the scenes are randomly placed, we first fine-tune the base models using randomly rotated 3D assets to enhance the priors. The first-stage model is fine-tuned on 32 NVIDIA RTX 5090 GPUs for 150K steps with a batch size of 128. The second-stage model is fine-tuned on 32 NVIDIA RTX 5090 GPUs for 450K steps with a batch size of 128. We also fine-tune the mesh decoder and the 3D Gaussian Splatting decoder on 32 NVIDIA RTX 5090 GPUs for 80K steps with a batch size of 128. For all fine-tuning stages, we use the AdamW [3] optimizer with a learning rate of $1e-5$. The Coarse Structure Completer and the Fine Shape Refiner are trained separately on 32 NVIDIA RTX 5090 GPUs for 80K steps with a batch size of 128. Before training on the scene-level dataset, we first pre-train the models on an object-level dataset for 100K steps. The Occlusion-Aware 3D Texturer is trained separately on 32 NVIDIA RTX 5090 GPUs for 90K steps with a batch size of 128 on the scene-level dataset. The Coarse Structure Completer and the Fine Shape Refiner are separately trained on 32 NVIDIA RTX 5090 GPUs for 80K steps with a batch size of 128. Before training on the scene-level dataset, we first pre-train the models on object-level dataset for 100K steps. The Occlusion-Aware 3D Texturer is trained separately on 32 NVIDIA RTX 5090 GPUs for 90K steps with a batch size of 128 on scene-level dataset. We use AdamW [3] optimizer with a learning rate of $5e-5$. In addition to the standard flow matching loss, we apply our proposed alignment loss to the three models with weighting factors of 0.1, 0.5, and 0.5, respectively. During inference, we use the classifier-free guidance with a guidance strength of 5, and the sampling steps are set to 25.



(a) Handle geometry distortion.



(b) Handle mask error.

Figure 1. Visualization of our scheme handling initial geometry distortions and mask errors.

2. Robustness to input noise

Tolerance to initial distortion. Although the initial geometries are distorted, our ARSG-110K contains massive high-quality 3D GT as supervision samples, therefore, 3D-Fixer can learn how to generate plausible 3D assets. Furthermore, we introduce three designs to improve the model's ability. First, the ORFA strategy (Sec. 3.5 in the main paper) provides detailed supervision. Second, the mixed source of initial geometries (Sec. 1) augments the supervision samples with more diverse distortion patterns. Third, the depth-ratio embedding (Sec. 1) helps 3D-Fixer to handle distortions of varying degrees. To quantify the robustness, we linearly interpolate the predicted depth and GT depth with a coefficient α to mimic different distortions. We report object-level (O) Chamfer Distance (CD), F1-Score@0.1 (FS), and Bounding Box IoU in Tab. 1a. The metrics and Fig. 1a indicate 3D-Fixer's robustness to initial distortions.

Tolerance to mask errors. Our scheme is able to handle mask errors, where multiple instances are merged into one mask as shown in Tab. 1b.

Table 1. Quantitative comparisons on our testset. We report Object (O) level Chamfer Distance (CD) and F-Score (FS), along with Bounding Box IoU, to quantify the robustness of our scheme on geometry distortion. We also report the object-level rendering metrics on our testset to quantify the texture quality compared to Gen3DSR (G3D).

(a) Distortion robustness.

α	$CD_O\downarrow$	$FS_O\uparrow$	$IoU\uparrow$
0.2	**0.185**	**60.55**	**0.547**
0.6	0.188	58.95	0.528
1.0	0.197	57.85	0.519

(b) Rendering metrics.

Method	$FID_V/FID_U\downarrow$	$CLIP_V/CLIP_U\uparrow$
G3D [?]	102.72 / 119.22	80.70 / 77.71
Ours	**43.52 / 46.25**	**89.82 / 88.51**

Robustness to complex occlusion patterns. Complex occlusion patterns cause severe distortions to the initial geometries, but our dataset and Contextual Conditioning design (Sec. 3.3 in the main paper) enable our method to robustly handle complex occlusion. First, our dataset contains complex occlusion patterns. On our testset, 40.18% of instance masks have more than one 8-connected component, and 10.89% have more than four. Second, our module jointly processes fragmented geometry and global features as mentioned in Sec. 3.3 of the main paper, which enables 3D-Fixer to reason about relationships among multiple visible parts. As shown in Fig. 2, 3D-Fixer can extract reliable information from the fragmented geometry rather than fully trusting it.

3. Texture quality

To access the quality of the synthesized texture, we separately render three views for the visible (V) and unseen (U) region on our testset, and report FID and CLIP score in Tab. 1b. These metrics demonstrate the quality and semantic consistency of the generated textures.

4. More results

In this section and in our Supplementary Video, we present diverse visualizations across a variety of scenarios. As in Fig. 3, our approach produces high-quality 3D assets and accurately reconstructs the spatial layout of the scene. In contrast, Gen3DSR generates blurry geometric structures, while MIDI fails to recover an accurate spatial layout.

As in Fig. 4, we further evaluate our method on real-world indoor scenes from ScanNet [2]. Our approach generalizes well to real scenes and produces coherent and high-quality 3D assets with accurate layout. However, Gen3DSR yields fragmented geometry and MIDI struggles to generate the accurate spatial layout. Furthermore, We report the real-world performance in Tab. 2c in the main paper on ScanNet dataset, where we use the following subset for evaluation: frame 360 from scene0048_00, frame 680 from scene0036_00, frame 105 from scene0033_00, frame 160 from scene0031_00, frame 440 from scene0028_00, frame 248 from scene0053_00, frame 235 from scene0087_00, frame 210 from scene0081_00, frame 105 from

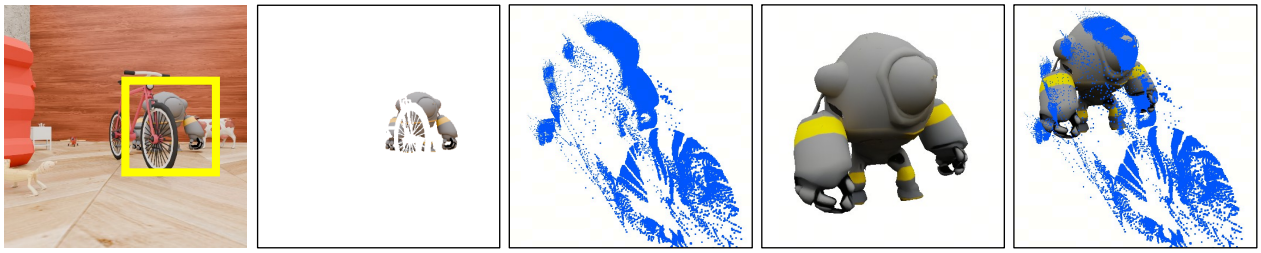


Figure 2. Visualization of our method handling complex occlusion patterns.

scene0199_00, frame 420 from scene0160_00, frame 200 from scene0162_00, frame 60 from scene0165_00, frame 1060 from scene0148_00, frame 940 from scene0134_00, and frame 1441 from scene0129_00.

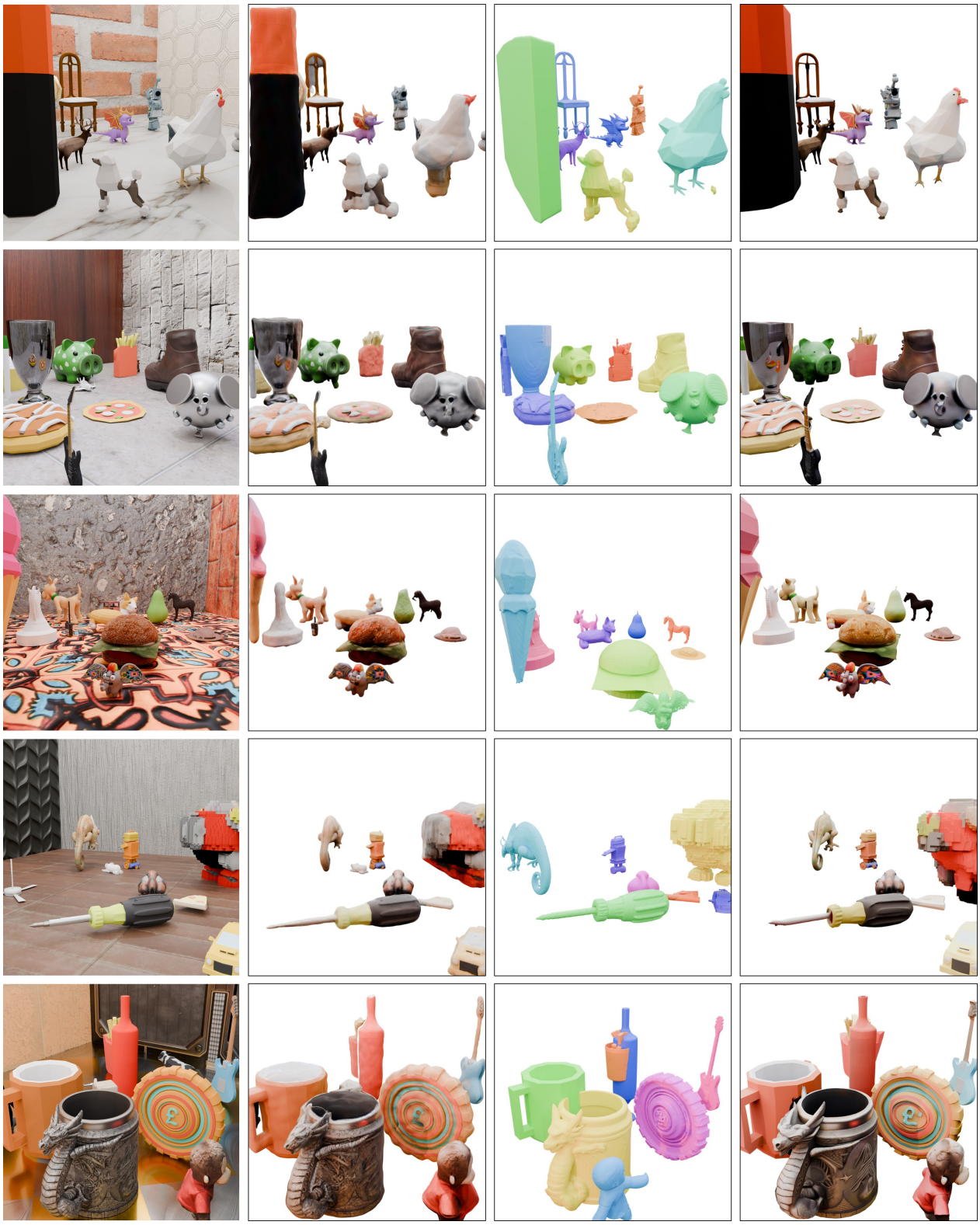
As in Fig. 5, we further evaluate our method on more challenging real captured scenes. Even in scenarios with complex layouts or a large number of instances, our method successfully generates accurate spatial arrangements and geometry, In contrast, MIDI encounters out-of-memory failures on an NVIDIA RTX 4090 GPU with 24 GB memory when dealing with scenes with large amounts of instances, and Gen3DSR generates fragmented or low quality geometries.

5. Procedural scene generation

To procedurally construct the ARSG-110K dataset, we use a subset of 180K high-quality 3D object assets from TRELLIS-500K [8]. To improve rendering photorealism, we additionally collect over 1K HDR maps and 5K material textures from BlenderKit, a community platform for sharing 3D assets. All scenes are rendered using the Blender Cycles engine. For each scene, we first create a floor plane, and then probabilistically place 0 to 4 additional planes around it as walls to simulate both indoor and outdoor environments. A material texture is randomly assigned to each plane. We also randomly select an HDR map for scene illumination. For object placement, we randomly sample 20 3D assets from the object pool, normalize each instance, apply a random rotation around the z-axis and a random scaling factor within $[0.5, 2.0]$, and then place the instances into the scene sequentially. To avoid interpenetration, each object is placed with at most 100 attempts. In each attempt, random scaling and rotation are applied, followed by collision detection against previously placed objects. The placement process terminates when a collision-free configuration is found or the maximum number of attempts is reached. The dataset and the scene construction script will be made publicly available.

6. Limitations and Future Works

As our method performs in-place completion using geometry-based cues for scene generation, the accuracy of the recovered layout inherently depends on the quality of the initial estimated geometry. We believe an important future direction is to explore unified frameworks to simultaneously estimate the scene geometry and generate the complete 3D instances in the scene.

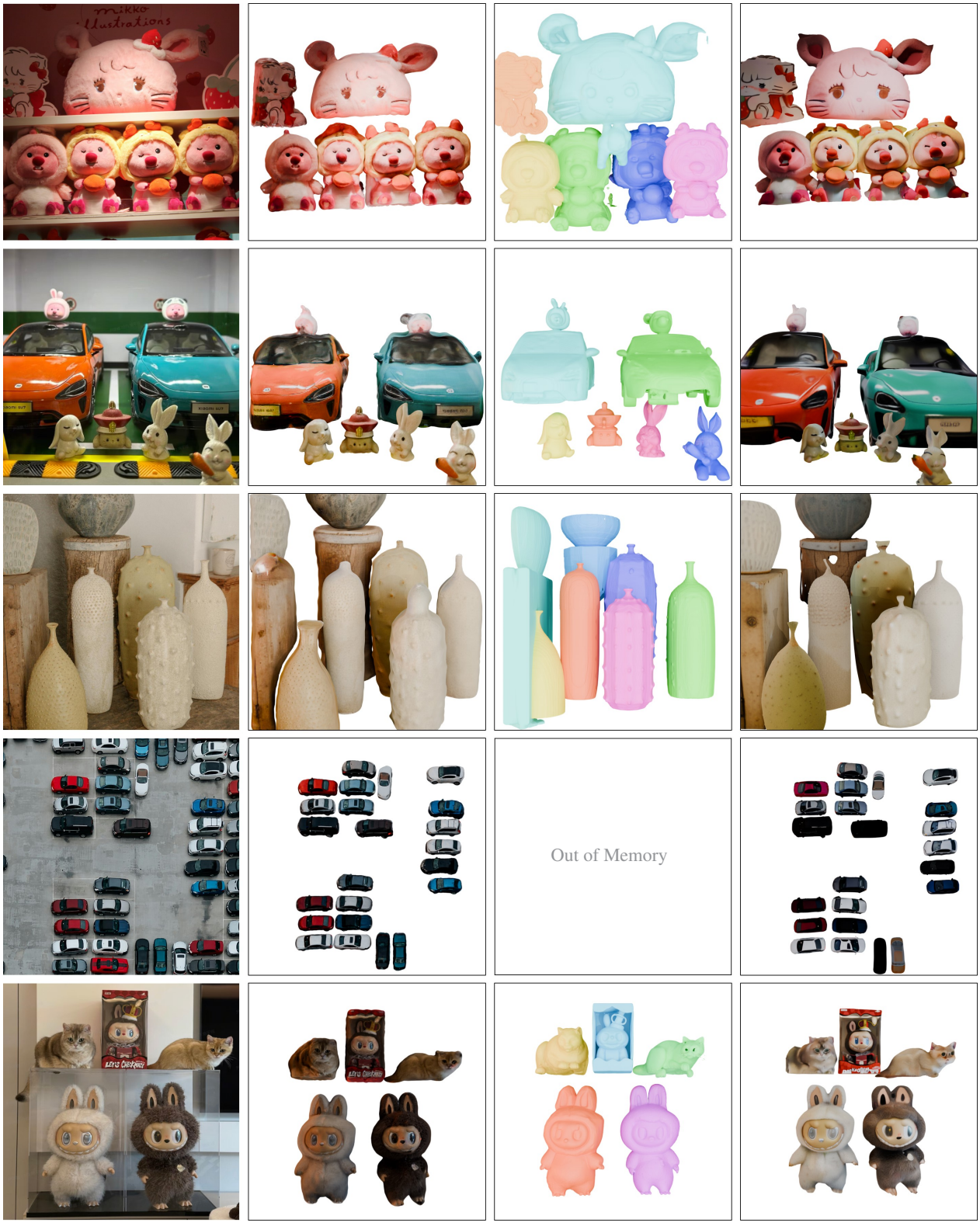


Input image Gen3DSR MIDI Ours

Figure 3. Visual comparisons on our testset.

Input image | Gen3DSR | MIDI | Ours

Figure 4. Visual comparisons on ScanNet.



Input image Gen3DSR MIDI Ours

Figure 5. Visual comparisons on real world captured images.

References

[1] Aleksei Bochkovskii, AmaÃĢl Delaunoy, Hugo Germain, Marcel Santos, Yichao Zhou, Stephan R Richter, and Vladlen Koltun. Depth pro: Sharp monocular metric depth in less than a second. *arXiv preprint arXiv:2410.02073*, 2024. 1

[2] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017. 2

[3] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 1

[4] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 1

[5] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023. 1

[6] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5294–5306, 2025. 1

[7] Ruicheng Wang, Sicheng Xu, Yue Dong, Yu Deng, Jianfeng Xiang, Zelong Lv, Guangzhong Sun, Xin Tong, and Jiaolong Yang. Moge-2: Accurate monocular geometry with metric scale and sharp details. *arXiv preprint arXiv:2507.02546*, 2025. 1

[8] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 21469–21480, 2025. 1, 3

[9] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *Advances in Neural Information Processing Systems*, 37: 21875–21911, 2024. 1