A. Implementation Details

A.1. Preliminaries: TRELLIS

We begin by briefly reviewing the TRELLIS [79] model on which our model is based. We chose TRELLIS because it is the state-of-the-art image-to-3D generator when we explored the task, and it is publicly available. However, our design is not specific to this model. TRELLIS is a conditional 3D diffusion model that performs denoising in a sparse 3D latent space. First, it introduces a transformer-based variational autoencoder (VAE) $(\mathcal{E}, \mathcal{D})$, where the encoder $\mathcal{E}$ maps sparse voxel features to structured latents $\boldsymbol{z}$, and the decoder $\mathcal{D}$ converts them into desired output representations, including 3D Gaussians [30], radiance fields, and meshes. In particular, a 3D object $\boldsymbol{o}_i$ is encoded using its *structured latent variables* (SLAT) defined as $\boldsymbol{z} = \{(\boldsymbol{z}_i, \boldsymbol{p}_i)\}_{i=1}^{L}$, where $\boldsymbol{z}_i \in \mathbb{R}^C$ is a local latent feature attached to the voxel at position $\boldsymbol{p}_i \in \{0, 1, \ldots, N-1\}^3$, N is the spatial resolution of the grid, and $L \ll N^3$ is the number of active voxels intersecting the object's surface. This representation encodes both coarse geometric structures and fine appearance details by associating local latents with active voxels.

TRELLIS comprises two diffusion models, one to predict the active voxel centers $\{\boldsymbol{p}_i\}_{i=1}^{L}$ (stage 1) and the other to recover the corresponding latents $\{\boldsymbol{z}_i\}_{i=1}^{L}$ (stage 2). Each model can be viewed as a denoising neural network v_θ operating in a latent space $\boldsymbol{\ell}$, and trained to remove Gaussian noise $\epsilon \sim \mathcal{N}(0, \boldsymbol{I})$ added to the latent code, *i.e.*, $\boldsymbol{\ell}^{(t)} = (1-t)\boldsymbol{\ell}^{(0)} + t\boldsymbol{\epsilon}$, where $t \in [0, 1]$ is the noise level [42]. The denoising network v_θ is trained to minimize the flow loss:

$$\min_{\theta} \mathbb{E}_{(\boldsymbol{\ell}^{(0)}, x), t, \boldsymbol{\epsilon}} \| v_\theta(\boldsymbol{\ell}^{(t)}, x, t) - (\boldsymbol{\epsilon} - \boldsymbol{\ell}^{(0)}) \|_2^2, \tag{A.1}$$

where x is the conditional input. In stage 1, the latent code is a compressed version of the $N \times N \times N$ occupancy volume, where the spatial resolution is reduced from $N = 64$ to $r = 16$. Hence, in this case, the latent vector is a matrix $\boldsymbol{\ell} \in \mathbb{R}^{L' \times C'}$ of $L' = r^3 = 4096$, C'-dimensional tokens. In stage 2, the latent code $\boldsymbol{\ell} = \{\boldsymbol{z}_i\}_{i=1}^{L} \in \mathbb{R}^{L \times C}$ is a matrix of L, C-dimensional tokens, where L is now the number of active voxels. Similar transformers are implemented to the corresponding denoising networks v_θ (Fig. 3). The conditioning image x is read via cross-attention layers that pool information from the tokens $\boldsymbol{c}_{\text{dino}}$ extracted by DINO [53] from the image x.

Given the active grid $\{\boldsymbol{p}_i\}_{i=1}^{K}$, the local latents $\{\boldsymbol{z}_i\}_{i=1}^{K}$ for each active voxel are predicted via iterative denoising using model (A.1), and these are subsequently decoded by the VAE to produce high-fidelity 3D outputs. The active grid $\{\boldsymbol{p}_i\}_{i=1}^{K}$, is obtained in a similar manner, denoising an (encoded and compressed versions of) the $N \times N \times N$ occupancy volume.

A.2. Network Design

We adopt the network design in TRELLIS [79] to load the pre-trained image-to-3D weights and integrate the mask-weighted cross-attention mechanism to each DiT block (24 blocks in total). And each image-conditioned cross-attention layer is immediately followed by an occlusion-aware cross-attention layer.

(a) Patchify and weight of visibility/occlusion mask The input condition image has a resolution of 512×512, which is resized to 518×518 to facilitate splitting into patches of size 14×14, as required by DINOv2 [52]. The resulting condition is then flattened into a tensor $\mathbf{c}_{dino} \in \mathbb{R}^{1374 \times 1024}$, where the sequence length corresponds to 37×37 patches plus 1 CLS token and 4 register tokens. To better align the visibility and occlusion masks with the DINOv2 features, we first split the masks into patches of the same size, then calculate the weight score for each patch using Eq. 4 and Eq. 5. The final $\mathbf{c}_{vis} \in \mathbb{R}^{1374 \times 1}$ and $\mathbf{c}_{occ} \in \mathbb{R}^{1374 \times 1}$ are obtained by flattening the weight scores, with a value of 1 assigned to the CLS and register token positions.

(b) Occlusion-aware cross-attention layer. We set the feature dimension of the occlusion-aware cross-attention layers to 1024, matching that of the image-conditioned cross-attention layers. To maintain consistent dimensions, we replicate the flattened occlusion masks to form a tensor $\mathbf{c}_{occ\_stack} \in \mathbb{R}^{1374 \times 1024}$.

(c) Multi-head Cross-Attention. Our mask-weighted multi-head cross-attention (MHA) layer, which is implemented to encourage the model to focus its attention on the visible parts of the object, is an extension of the cross-attention described in the main paper. Specifically, H heads are run in parallel, resulting in H attention scores. For mask-weighted attention mechanism, we impose $\boldsymbol{c}_{\text{vis}}$ simultaneously to each head:

$$\boldsymbol{A}_h = \mathrm{softmax}(\boldsymbol{S}_h + \log \boldsymbol{c}_{\text{vis}}), \tag{A.2}$$

$$\mathrm{MHA} = [\boldsymbol{A}_1 \boldsymbol{v}; \boldsymbol{A}_2 \boldsymbol{v}; \ldots; \boldsymbol{A}_H \boldsymbol{v}] \tag{A.3}$$

A.3. Training Details

(a) Pre-trained model loading. While TRELLIS is split into multiple modules, in our work we only train the sparse structure flow transformer and the structured latent flow transformer (see the overview figure where the ”fire” symbols indicate the parts that are fine-tuned, and ”snowflake” symbols indicate that we directly use the pre-trained weights).

(b) Data Augmentation. As described in Sec. 3.3, we generate random masks during training for data augmentation. Specifically, we begin by randomly drawing 1 to 3 lines, circles, and ellipses in the mask image. Next, to ensure these regions connect — thereby better simulating real-world occlusions, where mask regions are typically not highly fragmented — we randomly add 3 to 7 rectangular regions that have undergone an expansion operation. This results in a stable masking of the objects in the training data. Example inputs are presented in Fig. A.1.

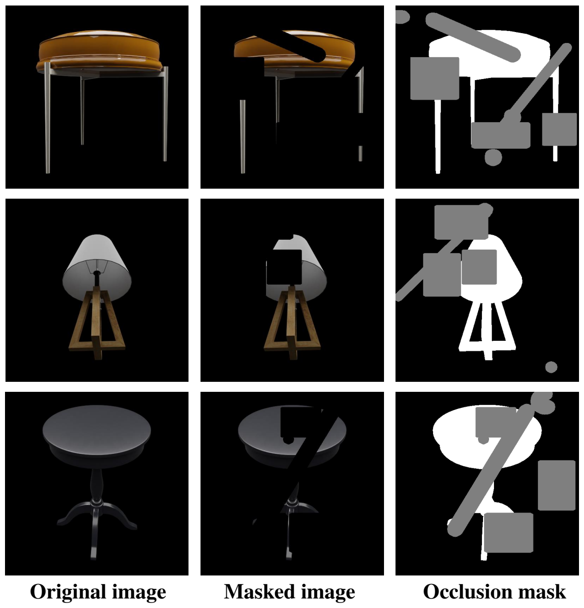

Figure A.1. **Examples of random mask generation.** The visible areas are shown in white, occluded areas in gray and background in black.

A.4. Inference Details

(a) 3D-consistent mask ratio. For the multi-view 3D-consistent masks described in Sec. 3.3, we set the mask ratio to a random number between 0.4 and 0.6 for each object, which results in a variety of reasonable mask areas.

(b) Time consumption. Despite the introduction of additional cross-attention layers, our inference time remains comparable to that of the baselines. Amodal3R can generate and render each object in under 10 seconds.

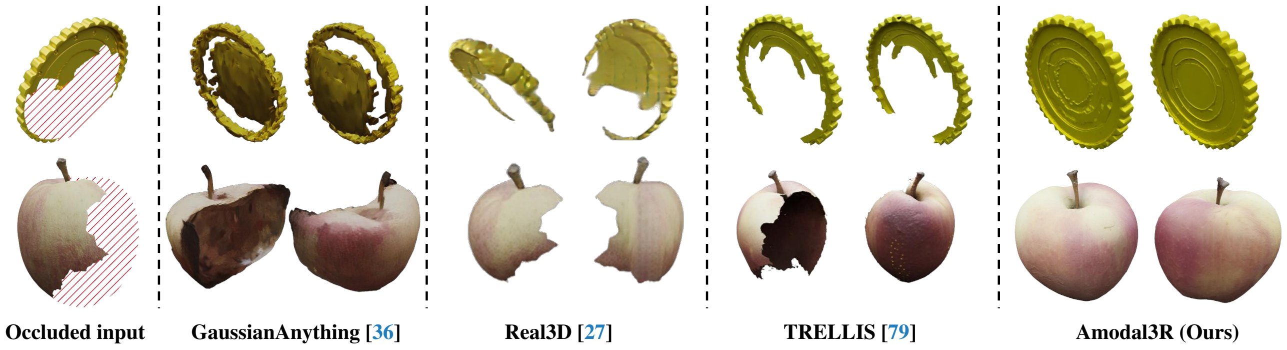

Figure A.2. **Examples using occluded images directly as the input of baseline models.**

B. Experimental Details

B.1. Evaluation Protocol

We evaluate the results using Google Scanned Objects (GSO) (1,030 objects) [16] and a randomly sampled subset of Toys4K [67] containing 1,500 objects. Here, we provide additional details regarding the computation of our evaluation metrics.

(a) Rendering quality and semantic consistency alignment To assess overall rendering quality, we compute the Fréchet Inception Distance (FID) [21] and Kernel Inception Distance (KID) [3]. Moreover, we evaluate semantic consistency using the CLIP score [57] by measuring the cosine similarity between the CLIP features of each generated image and its corresponding ground truth. For each object, we render 4 views using cameras with yaw angles of $\{0°, 90°, 180°, 270°\}$ and a pitch angle of 30°. The camera is positioned with a radius of 2.0 and looks at the origin with a FoV of 40°, consistent with TRELLIS [79]. While FID and KID are calculated between the ground truth and generated sets (6,000 images for Toys4K and 4,120 images for GSO), the CLIP score is calculated in a pair-wise manner, and we report the mean value to evaluate semantic consistency.

(b) Geometry quality For 3D geometry evaluation, we adopt Point cloud FID (P-FID) [49], Coverage Score (COV), and Minimum Matching Distance (MMD) using Chamfer Distance (CD). Following previous work [36, 49, 79], we sampled 4096 points from each GT/generated point cloud, which are obtained from the unprojected multi-view depth maps using the farthest point sampling.

C. More Results

In this section, we provide additional qualitative examples and comparison results to further demonstrate the performance of our Amodal3R.

C.1. Baselines using occluded input

We have stated in the main paper that ”occluded images will lead to incomplete or broken structures” in current 3D generative models. Here, we provide examples where pix2gestalt is omitted and the occluded images are directly used as the input. As shown in Fig. A.2, when baseline models receive images of partially visible objects as input, they often fail to recover complete and intact 3D assets.

C.2. More single-view to 3D examples

Due to the page restriction, we only provide limited examples in the main paper. Here we visualize more examples of single-view to 3D to further demonstrate the effectiveness of our method in Fig. D.4. The results show that compared with the 2D amodal completion + 3D generation baselines, our Amodal3R yields higher quality 3D assets across multiple categories.

C.3. More multi-view to 3D examples

We first provide visualized examples to explicitly explain the difference between the “pix2gestalt” and “pix2gestalt + MV” settings in the multi-view to 3D generation in Fig. C.3. For the “pix2gestalt” setting, we directly implement pix2gestalt for the amodal completion independently under each view. For the “pix2gestalt + MV” setting, we first choose the view with the greatest visibility from the 4 occluded views, then use pix2gestalt to complete the object (which is shown in the pix2gestalt column in the qualitative result), and subsequently use Zero123++ to get the 4 consistent views as the input of LaRa and TRELLIS. It can be observed that pix2gestalt alone results in obvious multi-view inconsistency, while with Zero123++ the consistency is significantly improved, thus leading to better 3D generation quality.

More multi-view to 3D examples are provided in Fig. D.5, where our Amodal3R again generates 3D assets with better quality than the baselines.

D. Discussion, Limitation, and Future Work

While Amodal3R achieves impressive 3D amodal completion, it comes with several limitations we hope to solve in the future. **1) Dataset expansion.** Due to the computational resources limitation, Amodal3R is trained on very limited data, consisting of only 20,627 synthetic 3D assets, predominantly confined to the furniture category. Consequently, completions on some complex or out-of-distribution objects may fail or lead to unrealistic structures. We believe that training on larger datasets, e.g. Objaverse-XL [15], could mitigate these issues. **2) Real-World data adaptation**. Different from pix2gestalt, Amodal3R is trained exclusively on synthetic data. As a result, it cannot leverage environmental cues and must rely solely on the visible portions of occluded objects for reconstruction. Creating real-world 3D modal datasets will further enhance the

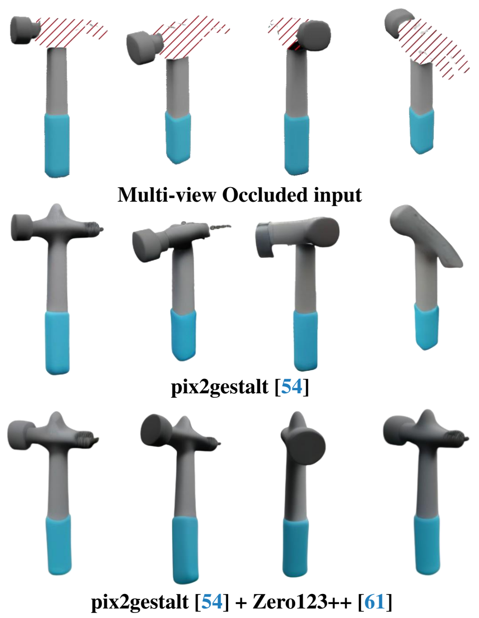

Figure C.3. **Example of "pix2gestalt" and "pix2gestalt + MV" input of multi-view to 3D evaluation.**

ability to apply models to real scenes. **3) Controllable completion**. Currently, how objects are completed is entirely up to the model itself and lacks control. Therefore, to further enhance the model to accept additional conditions, such as text, and allow users to edit and control the style of the completion process will be an important research direction for us in the future.

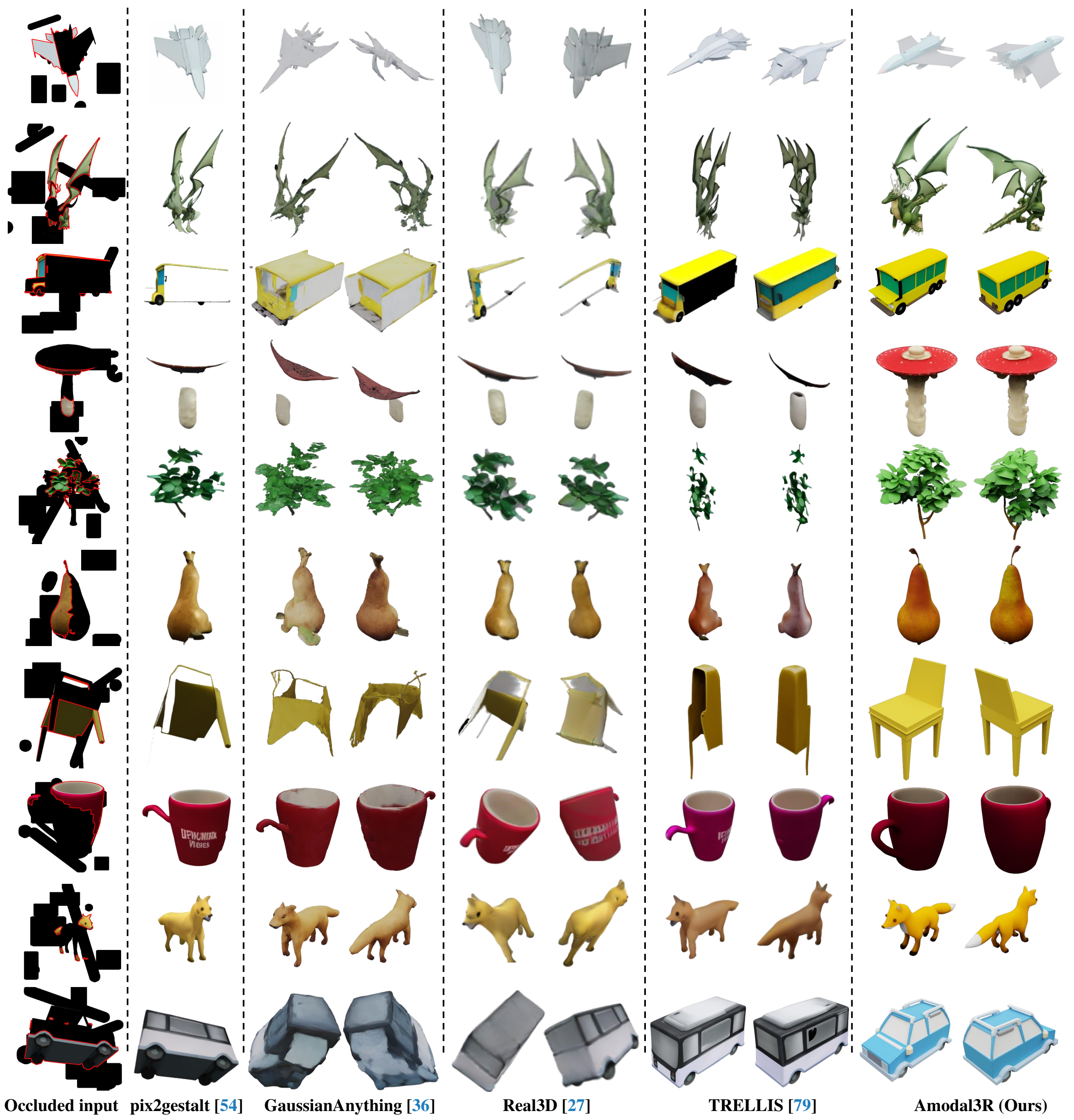

Figure D.4. **Additional single-view to 3D comparison examples.** The occluders are shown in black and the visible regions are highlighted with red outlines.

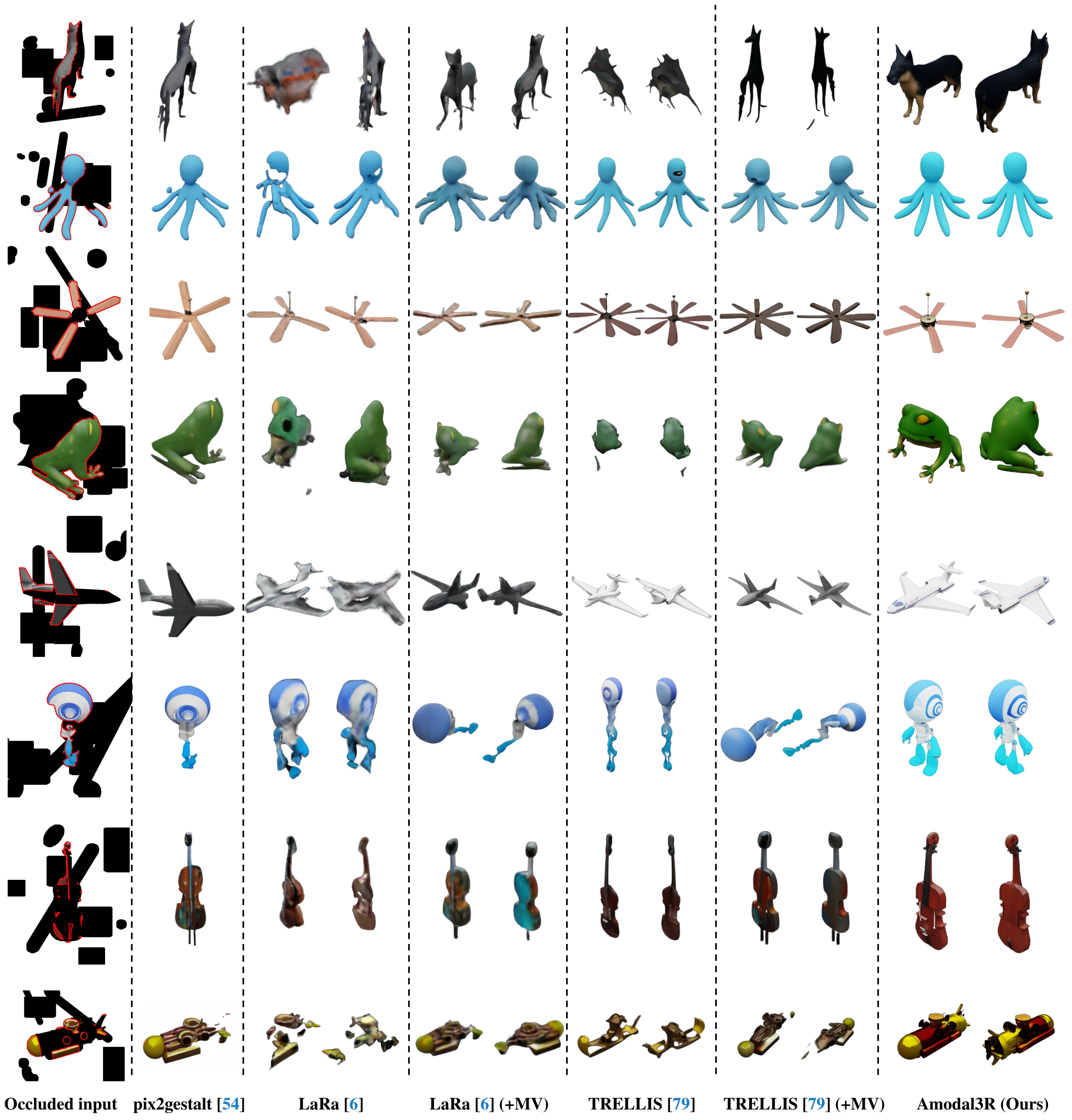

Figure D.5. **Additional multi-view to 3D comparison examples.** The occluders are shown in black and the visible regions are highlighted with red outlines.

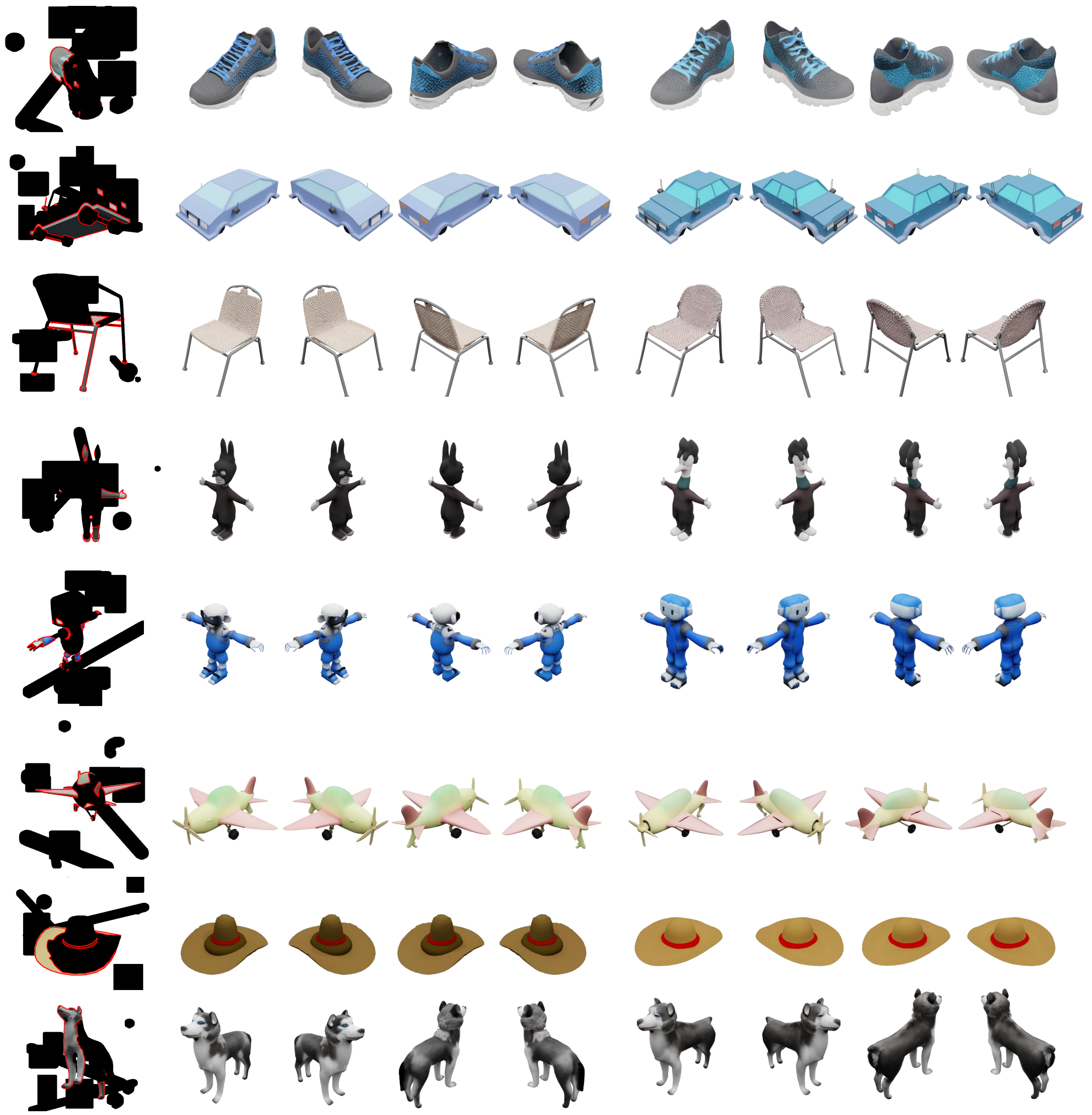

Figure D.6. **Additional diverse examples.** The occluders are shown in black and the visible regions are highlighted with red outlines.

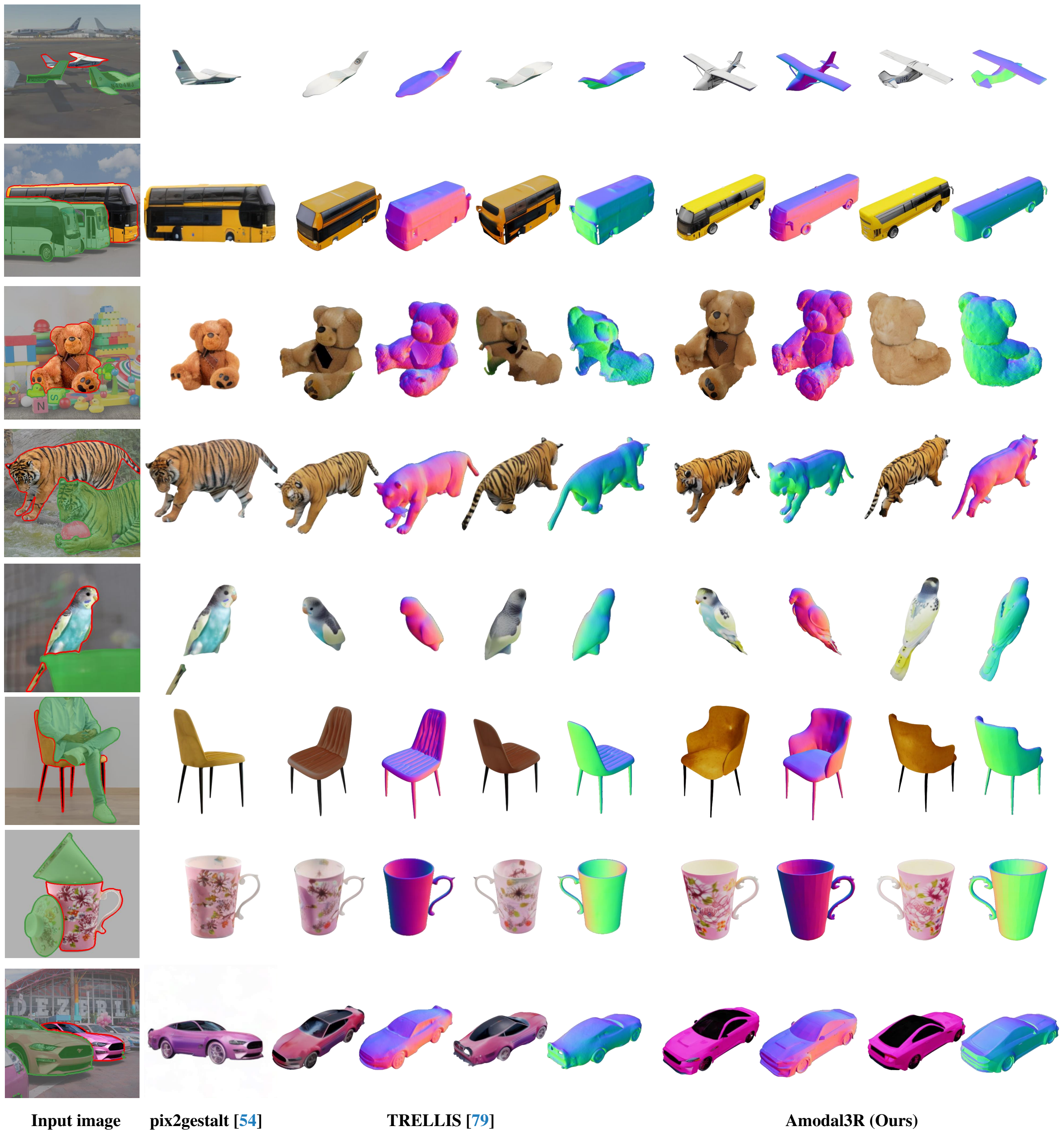

Figure D.7. **Additional in-the-wild examples compared with pix2gestalt + TRELLIS.** The target objects and occluders are marked with the red and green outlines.

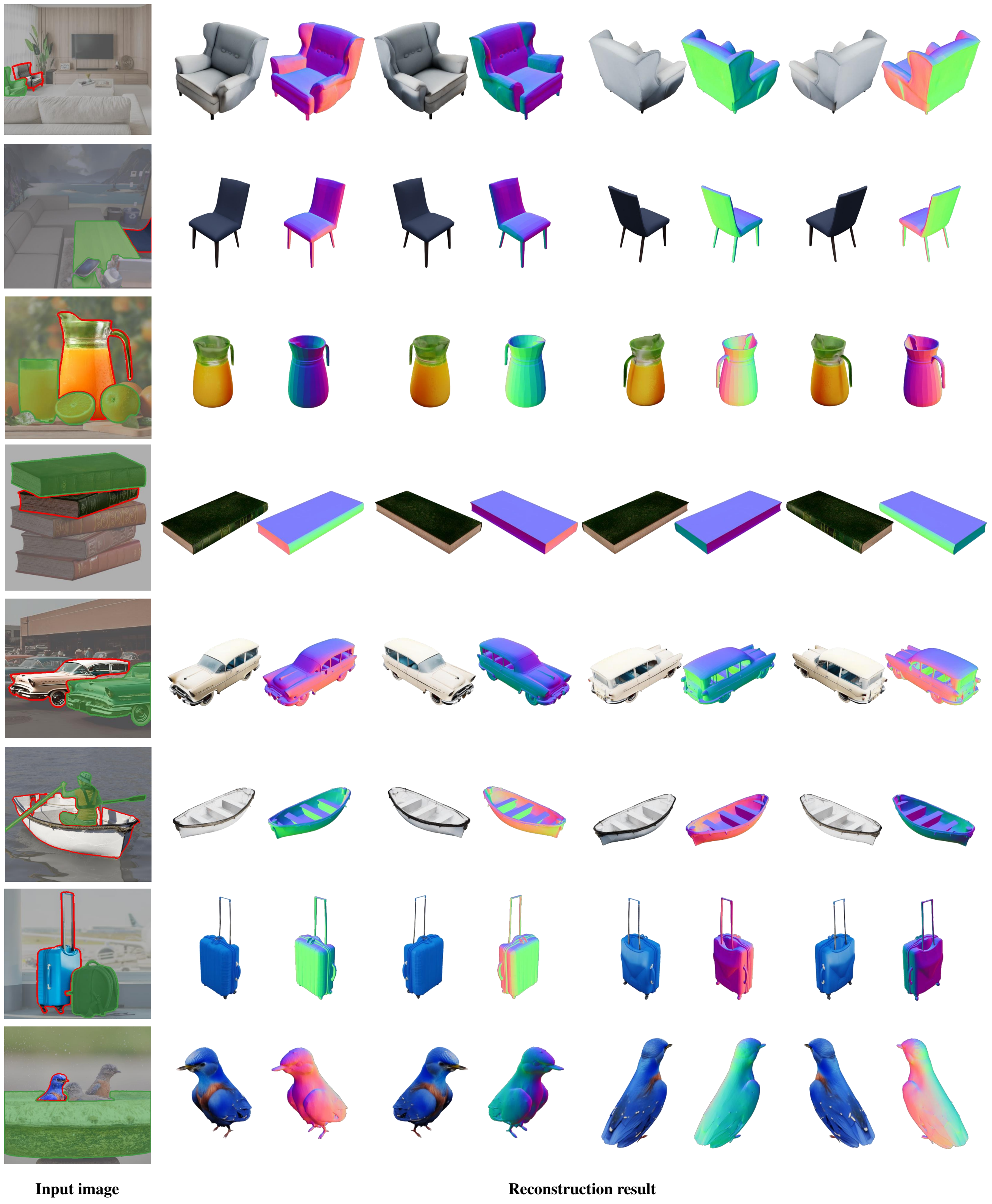

Figure D.8. **Additional in-the-wild examples.** The target objects and occluders are marked with the red and green outlines.

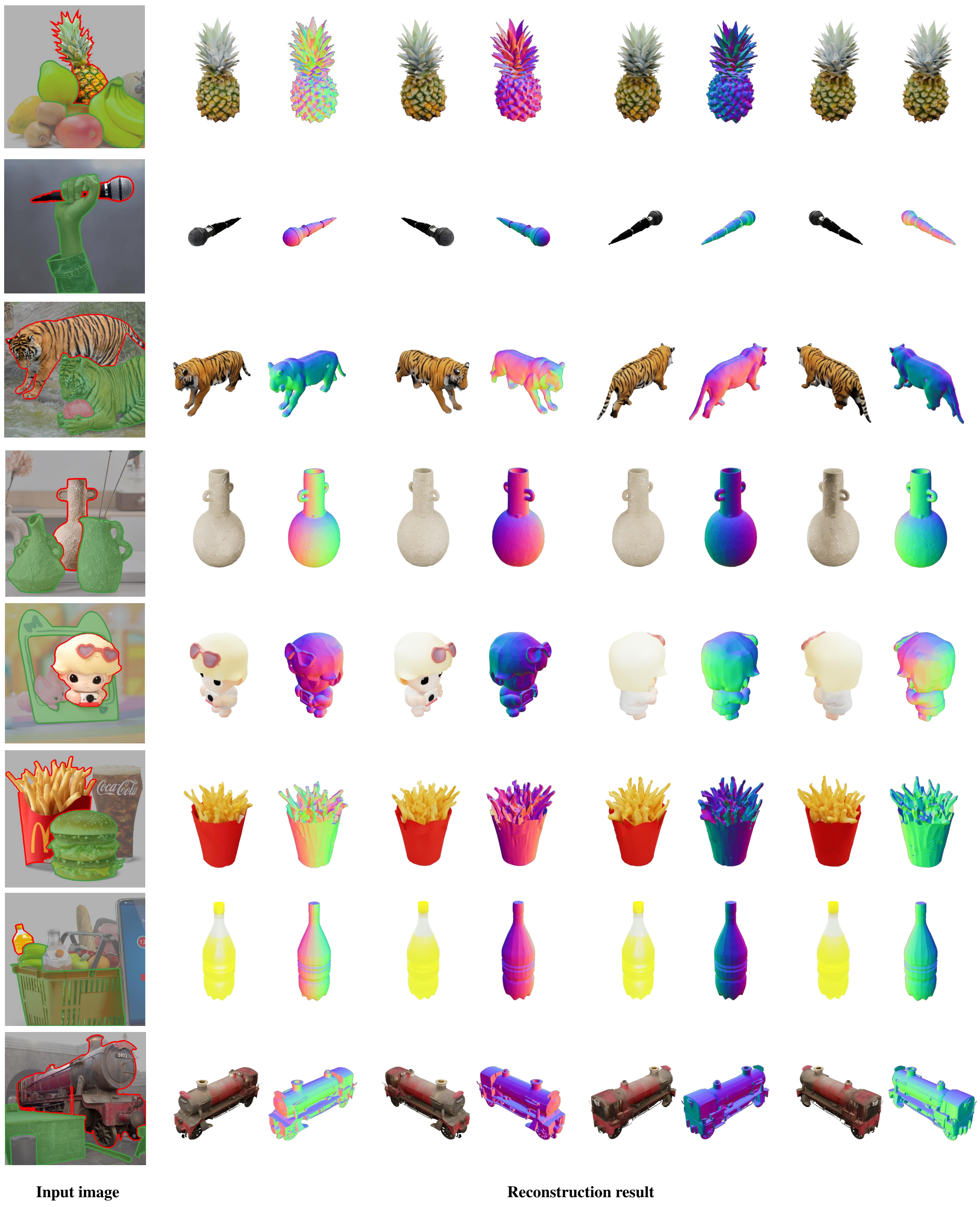

Figure D.9. **Additional in-the-wild examples.** The target objects and occluders are marked with the red and green outlines.