DetailGen3D: Generative 3D Geometry Enhancement via Data-Dependent Flow

Ken Deng[1,2,3] Yuanchen Guo[2] Jingxiang Sun[1] Zixin Zou[2] Yangguang Li[2] Xin Cai[4]
Yanpei Cao[2] Yebin Liu[1] Ding Liang[1,2]

[1]Tsinghua University [2]VAST [3]Sun Yat-sen University [4]The Chinese University of Hong Kong

Supplementary Material

In this supplement, we first provide the implementation detail in Sec. 1. We also provide further experiment details in Sec. 2. Finally, we provide additional visual results in Sec. 3. We encourage the readers to view our accompanying videos in the supplement, showcase the rotation of objects rendered with normals as presented in the paper.

1. Implementation Details

In our training setup, we start with a learning rate of 1e-10 and warm it up to 1e-4 over 5,000 steps. We use a total batch size of 256. We train our *DetailGen3D* model for 1,000 epochs, which takes approximately eight days on eight A800 GPUs. When training the data-dependent rectified flow, we randomly zero the DINO features with a probability of 10% to enable classifier-free guidance during inference, thereby improving the quality of conditional generation. For the DINO V2 [4] checkpoint, we use the ViT-L/14 distilled with the registered version, downloaded from the official DINO V2 GitHub repository[1].

At inference time, we extract tokens from the coarse geometry using FPS-VAE and, along with the image prompt's DINO feature, input them into DiT [5]. After denoising with a guidance scale of 4 over 10 sampling steps, we decode the predicted tokens using the FPS-VAE decoder to obtain refined geometry.

In our dataset design, we first normalize and rescale each object to fit within a bounding box of side length 1, then translate the object so that the bounding box is centered at the origin. To reduce storage overhead, we randomly sample 100,000 points from each mesh. For the FPS-VAE input point cloud, we further randomly subsample these points from 100,000 down to 20,480.

2. Experiment Details

For the feed-forward reconstruction and generation experiment, the FID [3] metric we evaluate following SDF-stylegan [7], rendering 20 views with random camera poses.

For the optimization based reconstruction refinement experiment, we render 40 views to train the neus. The camera poses are elevation with -60, -30, 0, 30, 60 and azimuth with 0, 45, 90, 135, 180, 225, 270, 315.

3. Additional Visual Results

We provide additional visual results in this section. Fig 1 shows GSO [2] generation results. Fig 2,3,4 shows Objaverse [1] generation results. Fig 5,6 shows GPTEval3D [6] generation results. Fig 9 shows GSO [6] generation results. Fig 10, 11, 12, 13, 14, shows Objaverse [1] generation results.

We also provide more ablation study visual results about noise augmentation, as presented in Fig 15, 16.

[1] https://github.com/facebookresearch/dinov2

Algorithm 1 Data-Dependent Rectified Flow

```
procedure $\mathcal{Z}$(RectFlow($(X_0, X_1)$))
    Inputs: Draws from a coupling $(X_0, X_1)$ of $\pi_0$ and $\pi_1$; velocity model $v_\theta : \mathbb{R}^d \to \mathbb{R}^d$ with parameter $\theta$.
    Training:
    1. Obtain $\pi_0$ and $\pi_1$ using $\pi_0 = \mathcal{E}(G_0)$ and $\pi_1 = \mathcal{E}(G_1)$, where $\mathcal{E}$ denotes the VAE encoder, $G_0$ represents the coarse geometry, and $G_1$ represents the fine geometry.
    2. Optimize $\hat{\theta} = \arg\min_\theta \mathbb{E}\left[\|X_1 - X_0 - v(tX_1 + (1-t)X_0, t)\|^2\right]$, with $t \sim$ Uniform$([0,1])$.
    Sampling:
    1. Start with $Z_0 \sim \pi_0$, where $\pi_0 = \mathcal{E}(G_0)$. Here, $\mathcal{E}$ denotes the VAE encoder, $G_0$ represents the coarse geometry.
    2. Generate $(Z_0, Z_1)$ by solving $\mathrm{d}Z_t = v_{\hat{\theta}}(Z_t, t)\mathrm{d}t$, obtaining $\{Z_t : t \in [0,1]\}$.
    Return: $\mathcal{Z} = \{Z_t : t \in [0,1]\}$.
end procedure
```

References

[1] 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, 2022. 1

[2] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B. McHugh, and Vincent Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items, 2022. 1

[3] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017. 1

[4] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2024. 1

[5] William Peebles and Saining Xie. Scalable diffusion models with transformers. *arXiv preprint arXiv:2212.09748*, 2022. 1

[6] Tong Wu, Guandao Yang, Zhibing Li, Kai Zhang, Ziwei Liu, Leonidas Guibas, Dahua Lin, and Gordon Wetzstein. Gpt-4v(ision) is a human-aligned evaluator for text-to-3d generation. In *CVPR*, 2024. 1

[7] Xin-Yang Zheng, Yang Liu, Peng-Shuai Wang, and Xin Tong. Sdf-stylegan: Implicit sdf-based stylegan for 3d shape generation. In *Comput. Graph. Forum (SGP)*, 2022. 1

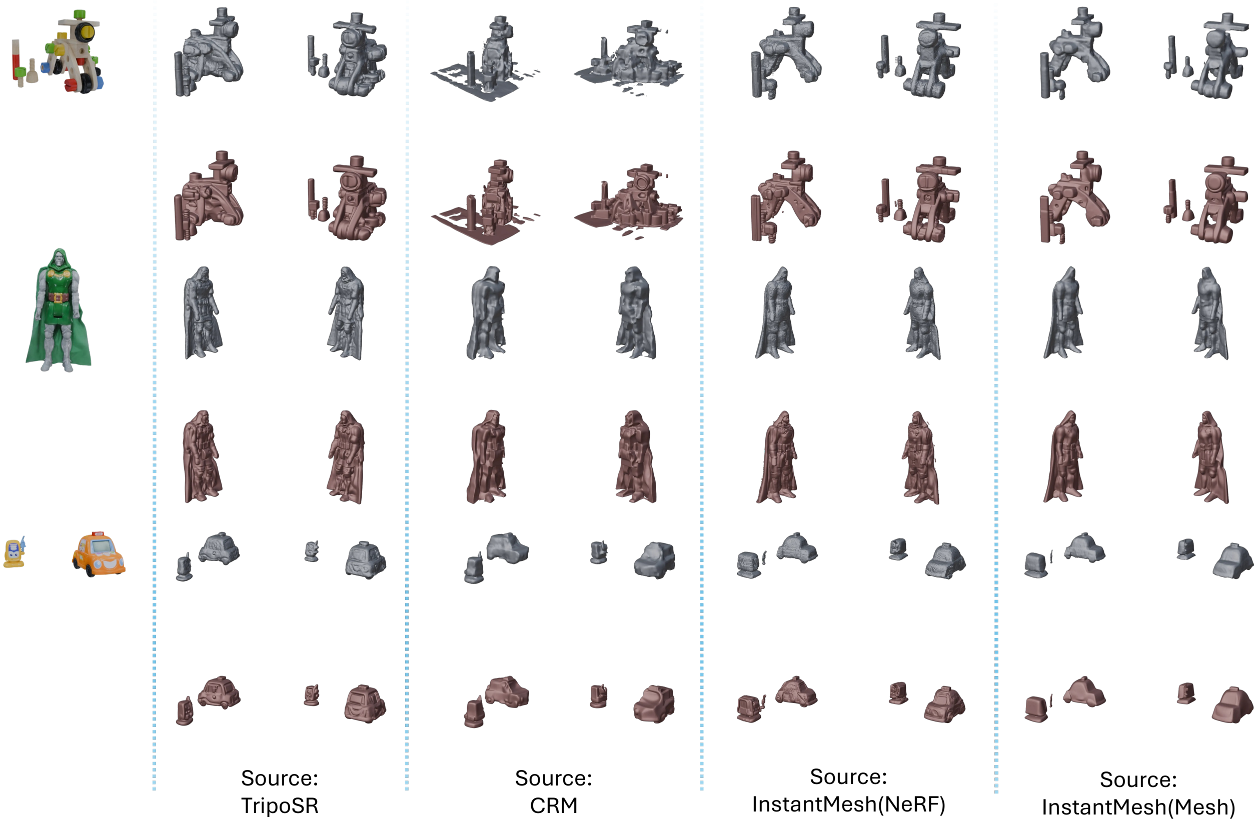

Figure 1. Generation results on GSO. ▨ represent coarse, 🟥 represent fine refinement results from our method.

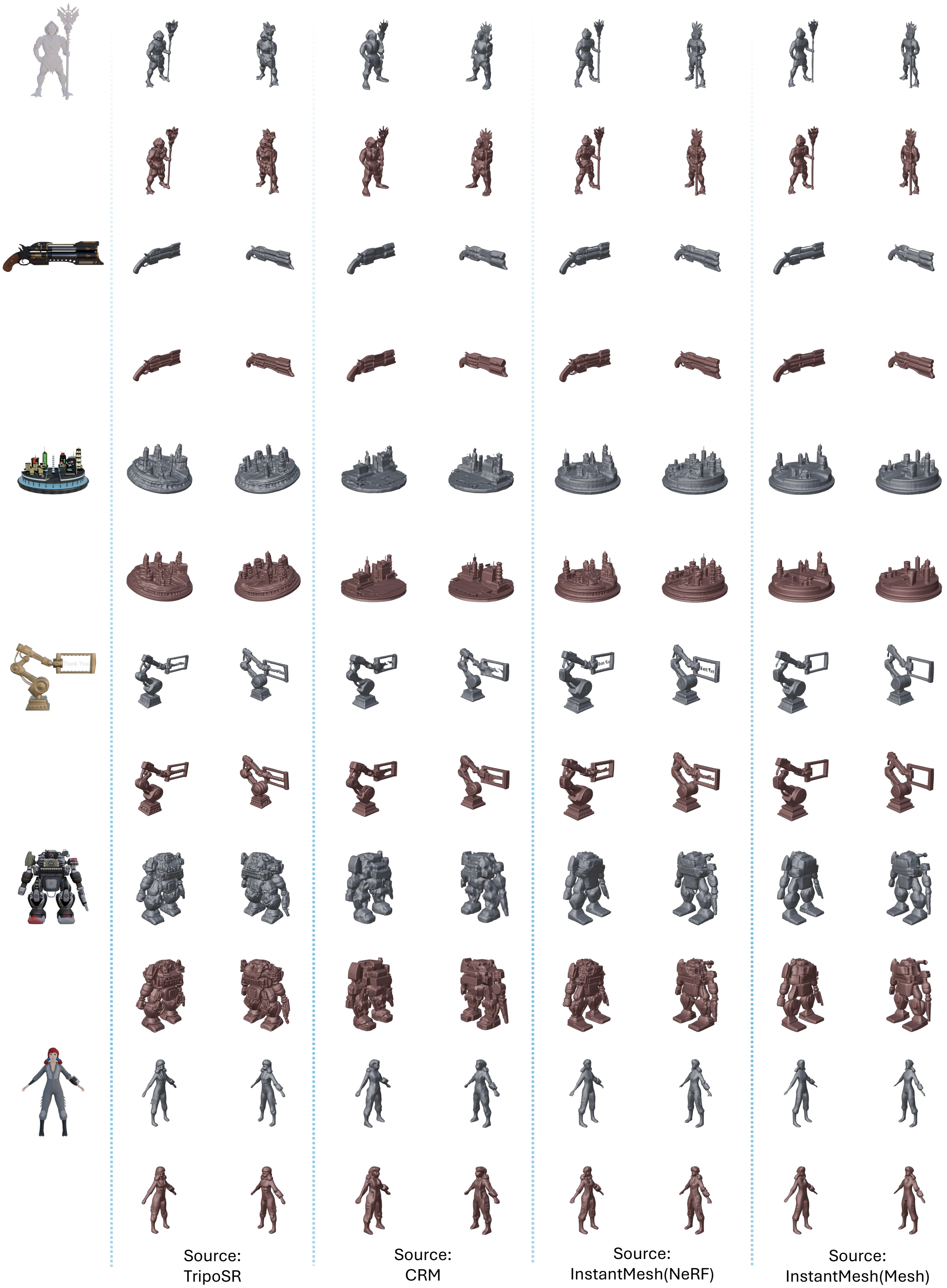

Figure 2. Generation results on Objaverse. 🟦 represent coarse, 🟥 represent fine refinement results from our method.

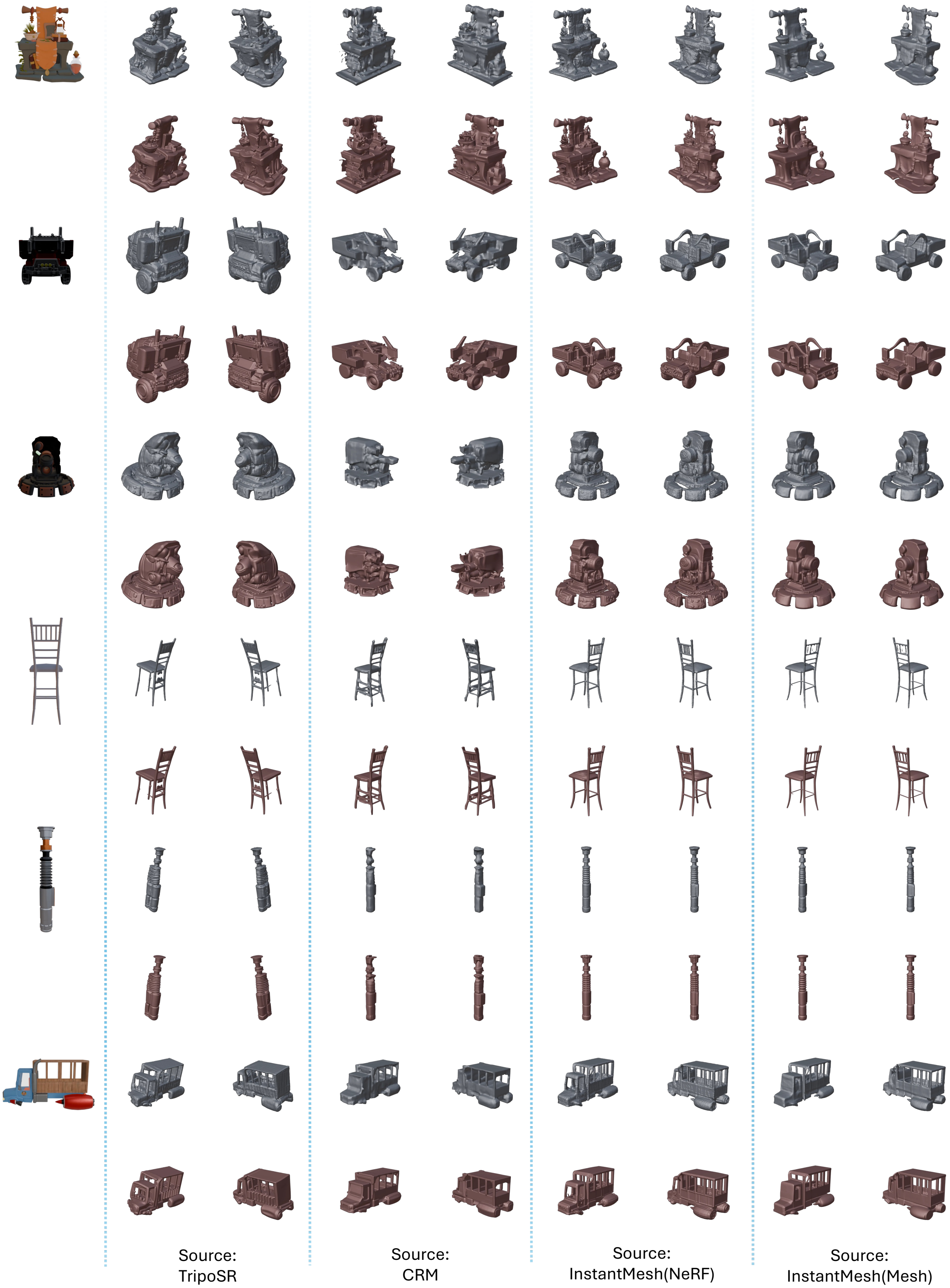

Figure 3. Generation results on Objaverse. 🟦 represent coarse, 🟥 represent fine refinement results from our method.

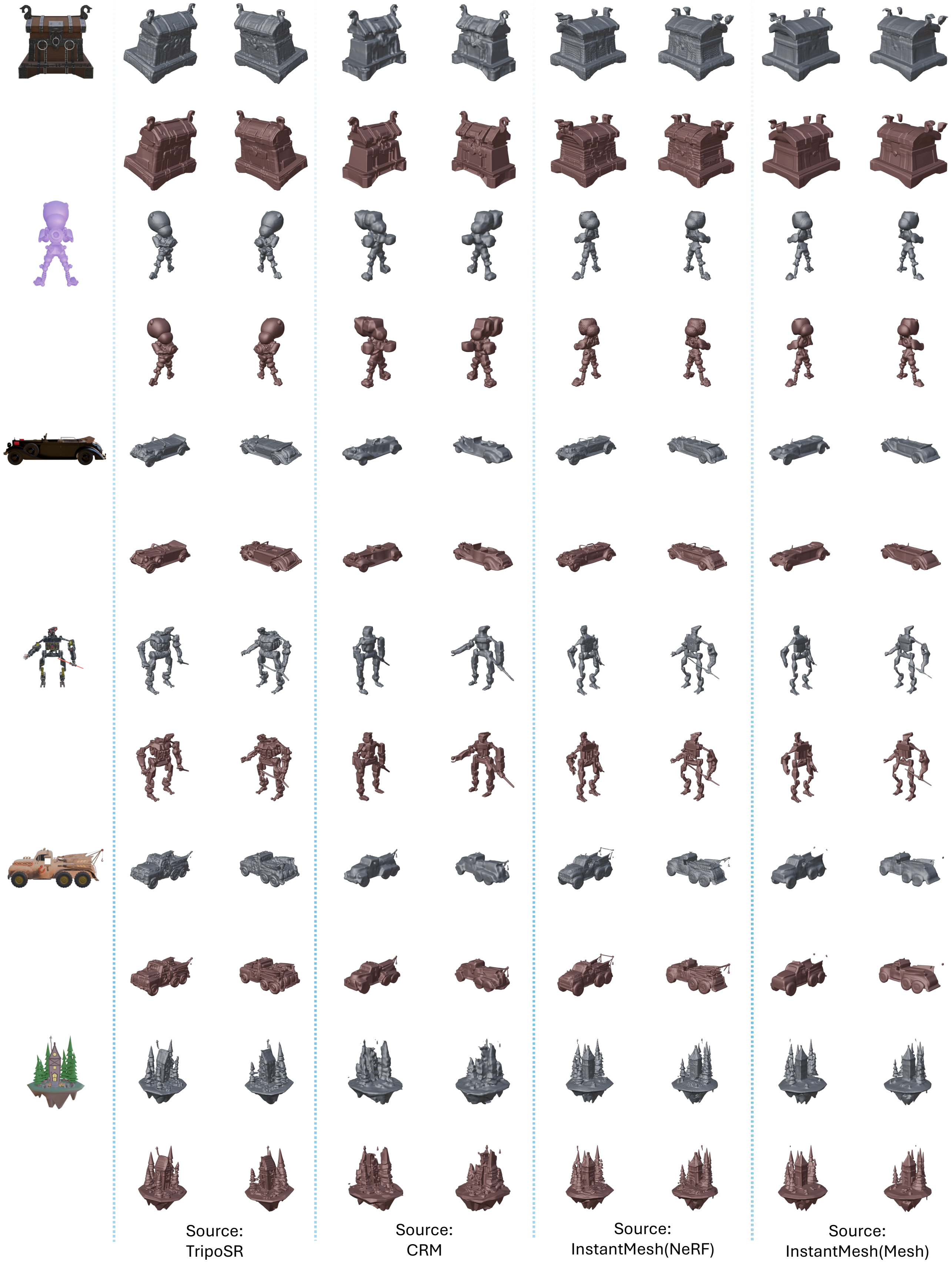

Figure 4. Generation results on Objaverse. ■ represent coarse, 🟥 represent fine refinement results from our method.

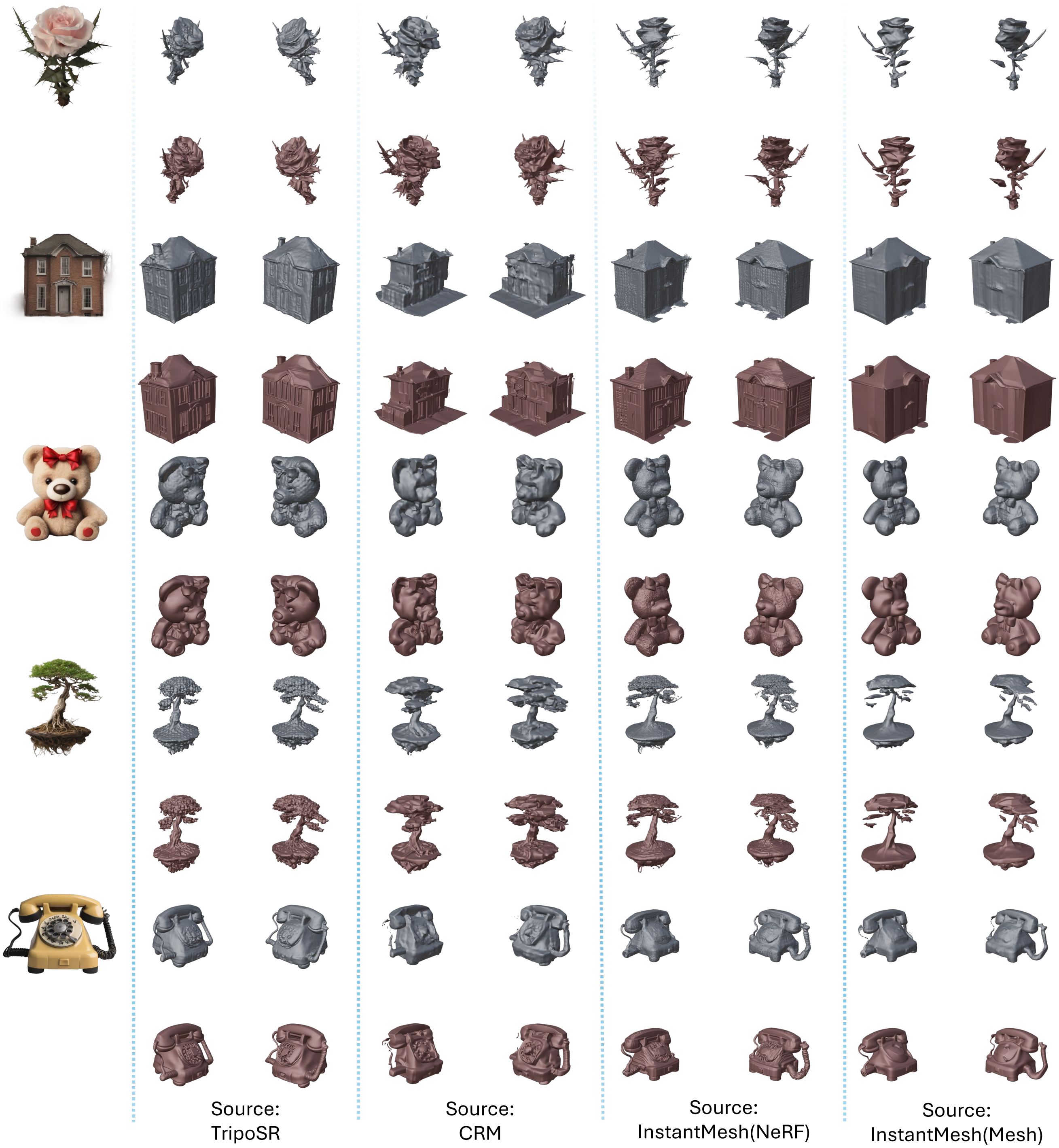

Figure 5. Generation results on GPTEval3D. 🟦 represent coarse, 🟥 represent fine refinement results from our method.

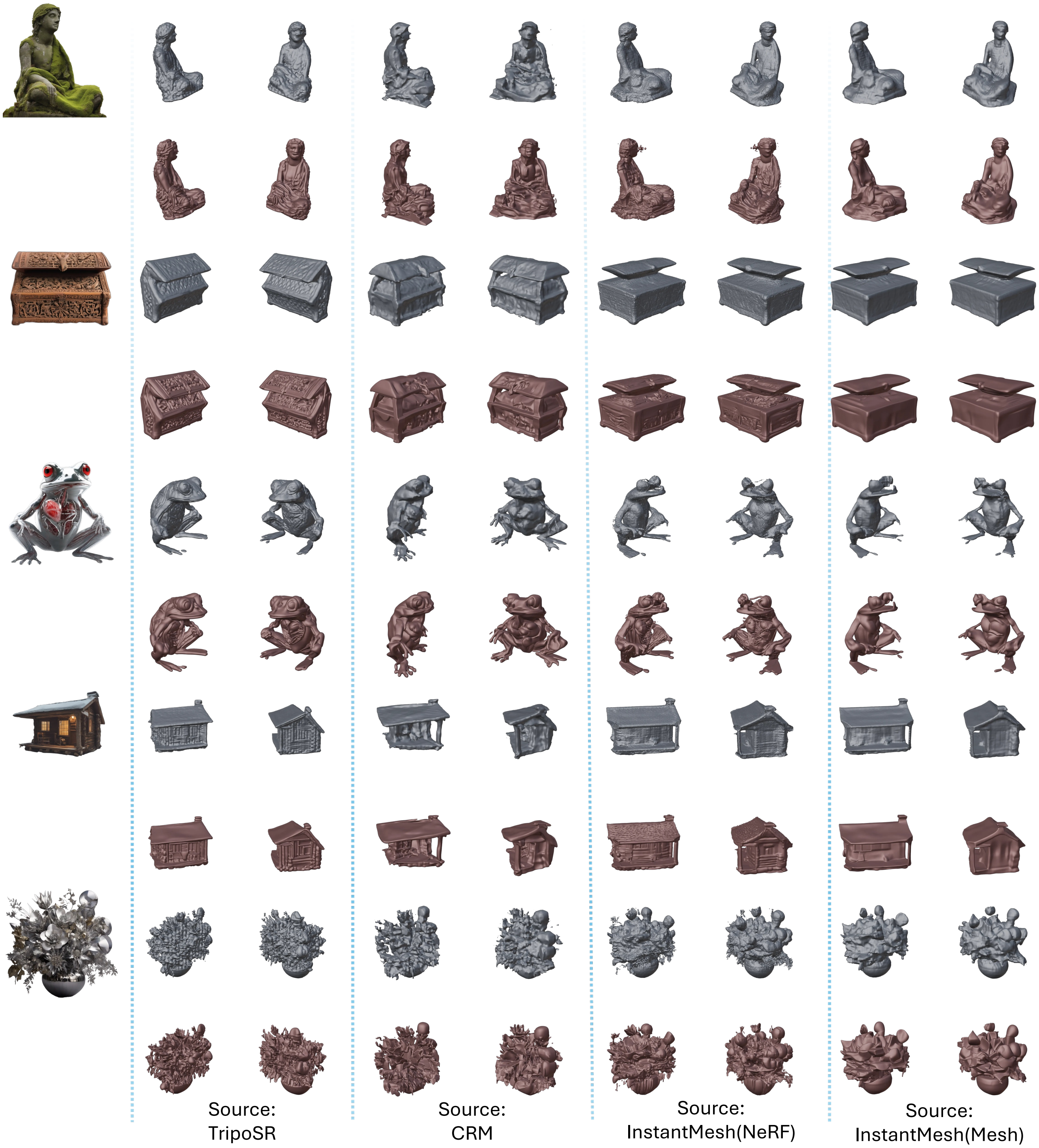

Figure 6. Generation results on GPTEval3D. ■ represent coarse, 🟥 represent fine refinement results from our method.

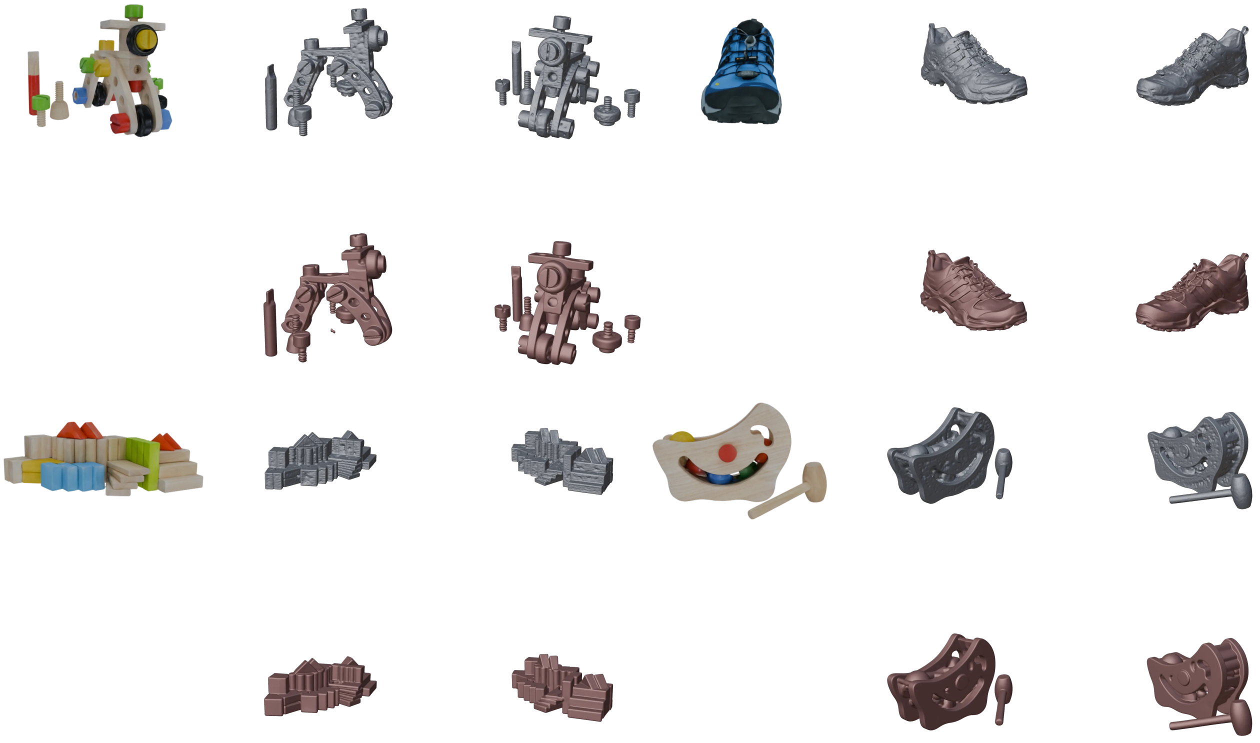

Figure 7. Optimization-based reconstruction results on GSO. ■ represent coarse, 🟥 represent fine refinement results from our method.

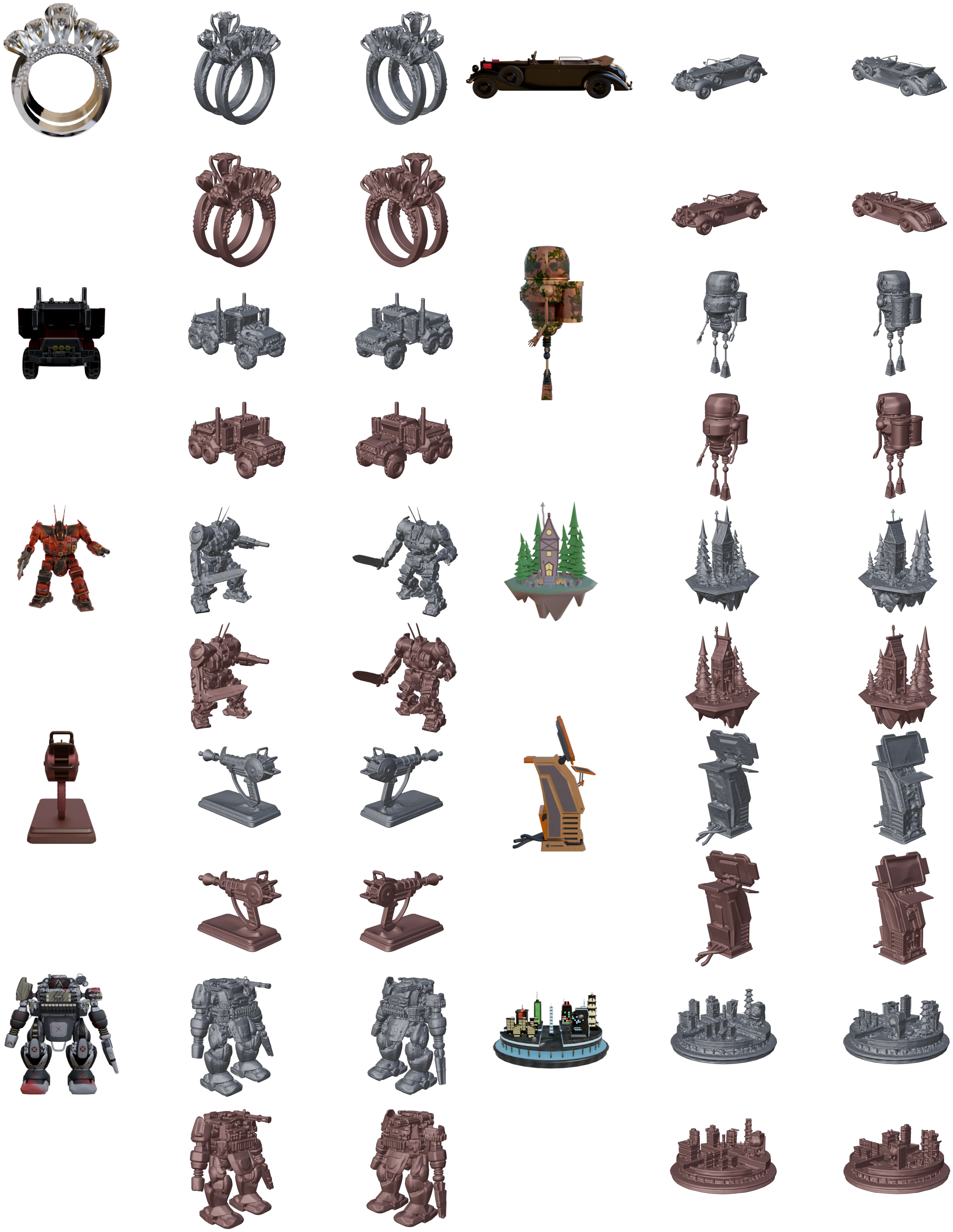

Figure 8. Optimization-based reconstruction results on Objaverse. 🟦 represent coarse, 🟥 represent fine refinement results from our method.

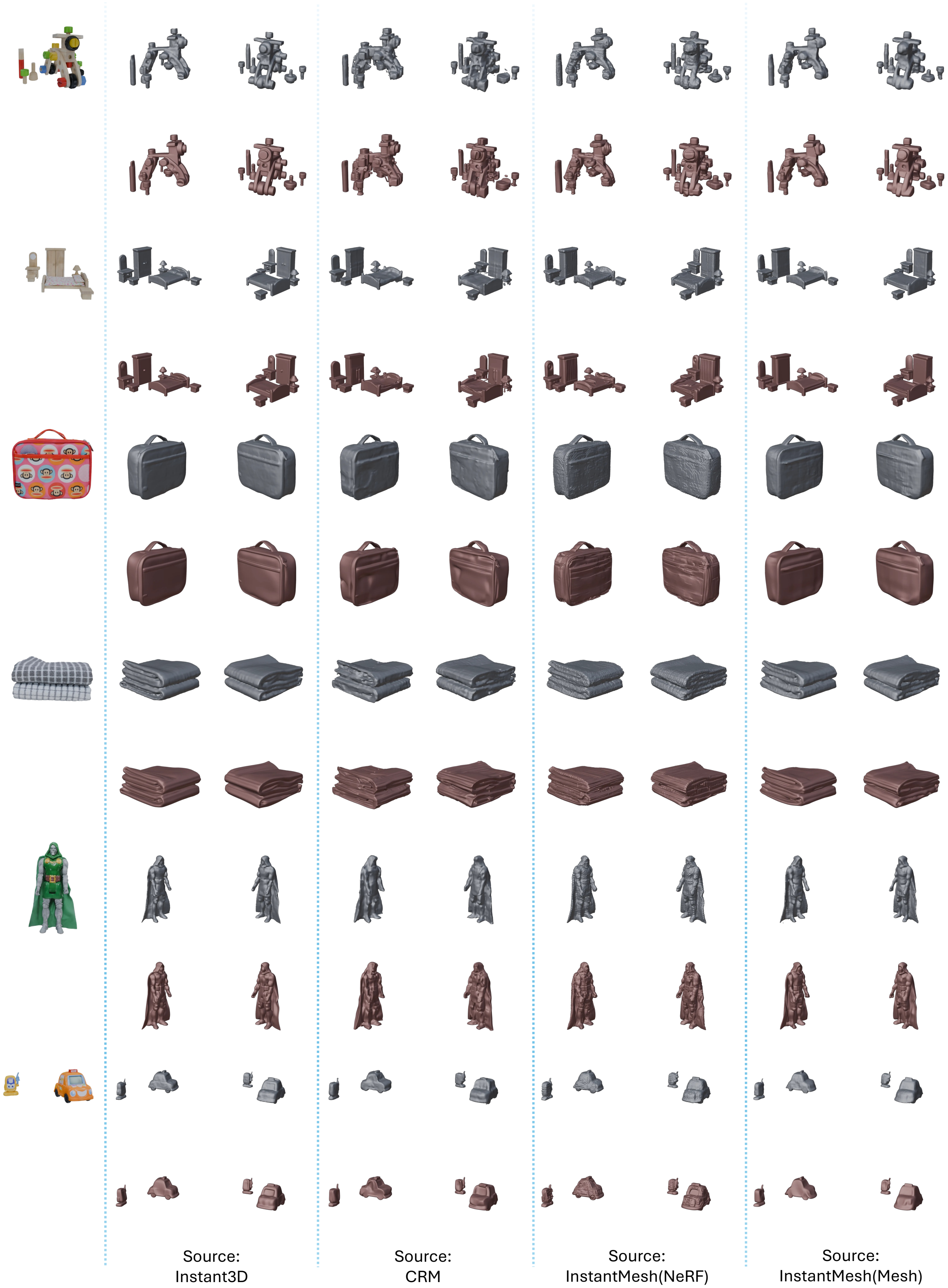

Figure 9. Feed-forward reconstruction results on GSO. ■ represent coarse, 🟥 represent fine refinement results from our method.

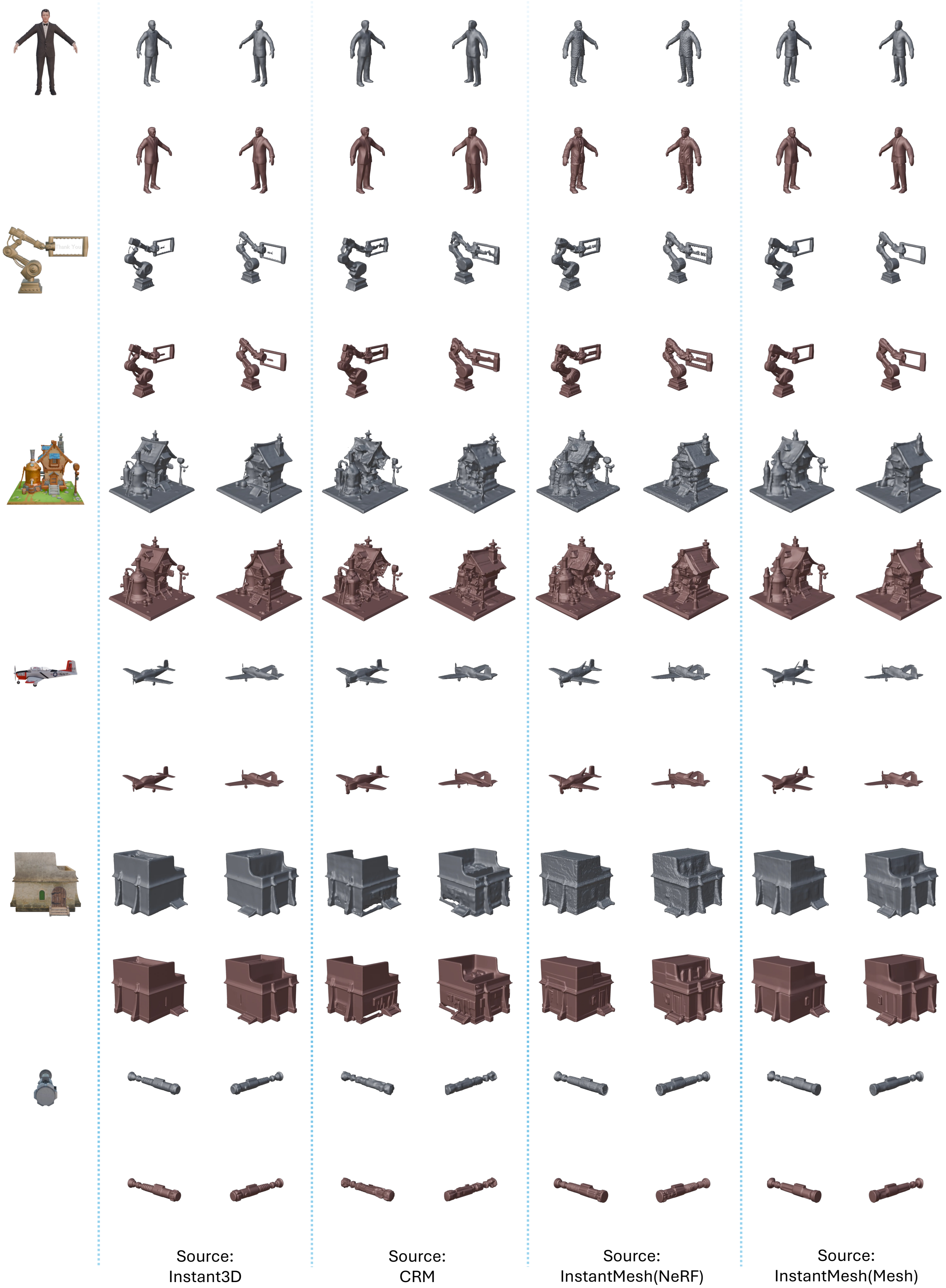

Figure 10. Feed-forward reconstruction results on Objaverse. 🟦 represent coarse, 🟥 represent fine refinement results from our method.

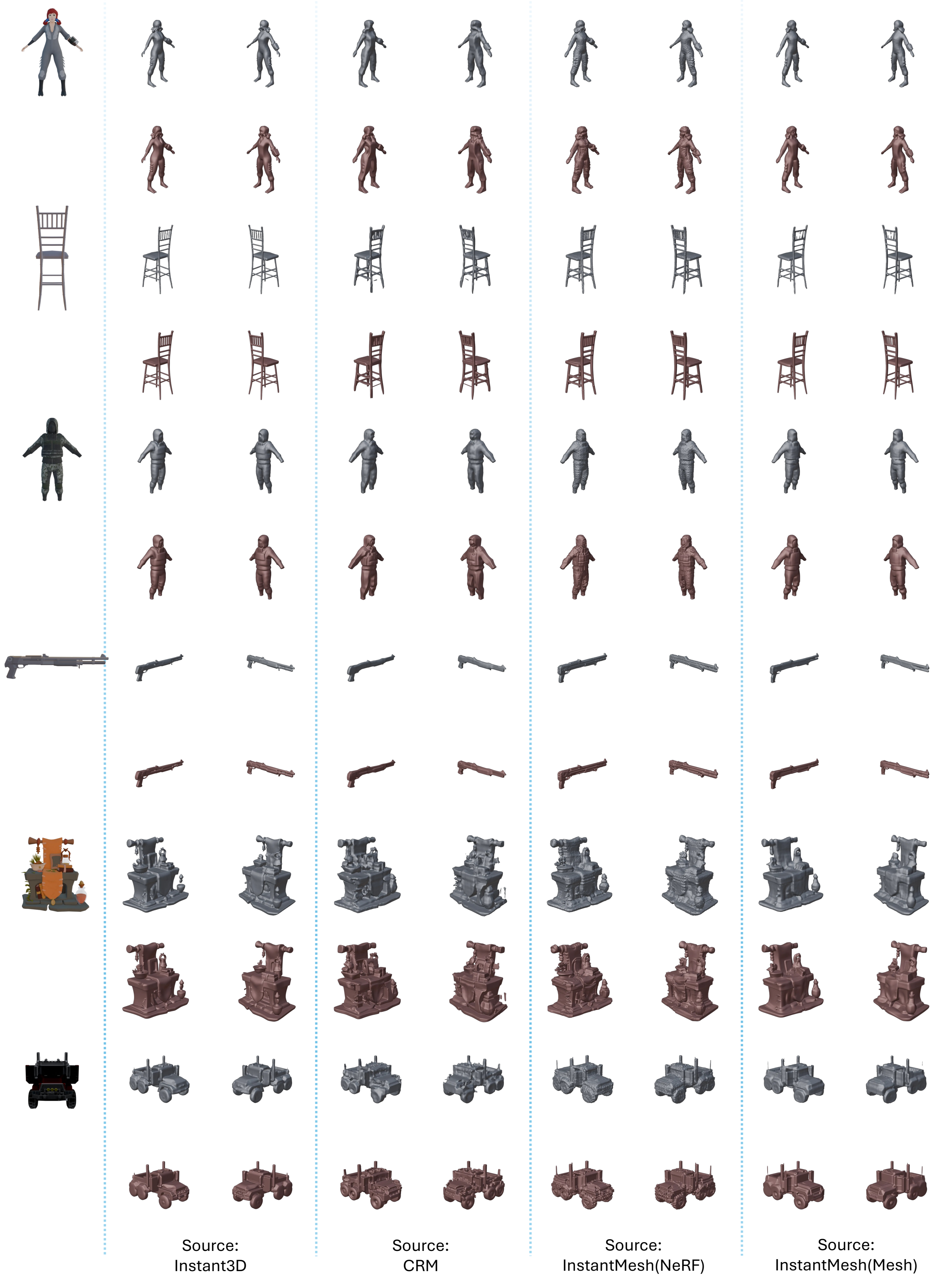

Figure 11. Feed-forward reconstruction results on Objaverse. ■ represent coarse, 🟥 represent fine refinement results from our method.

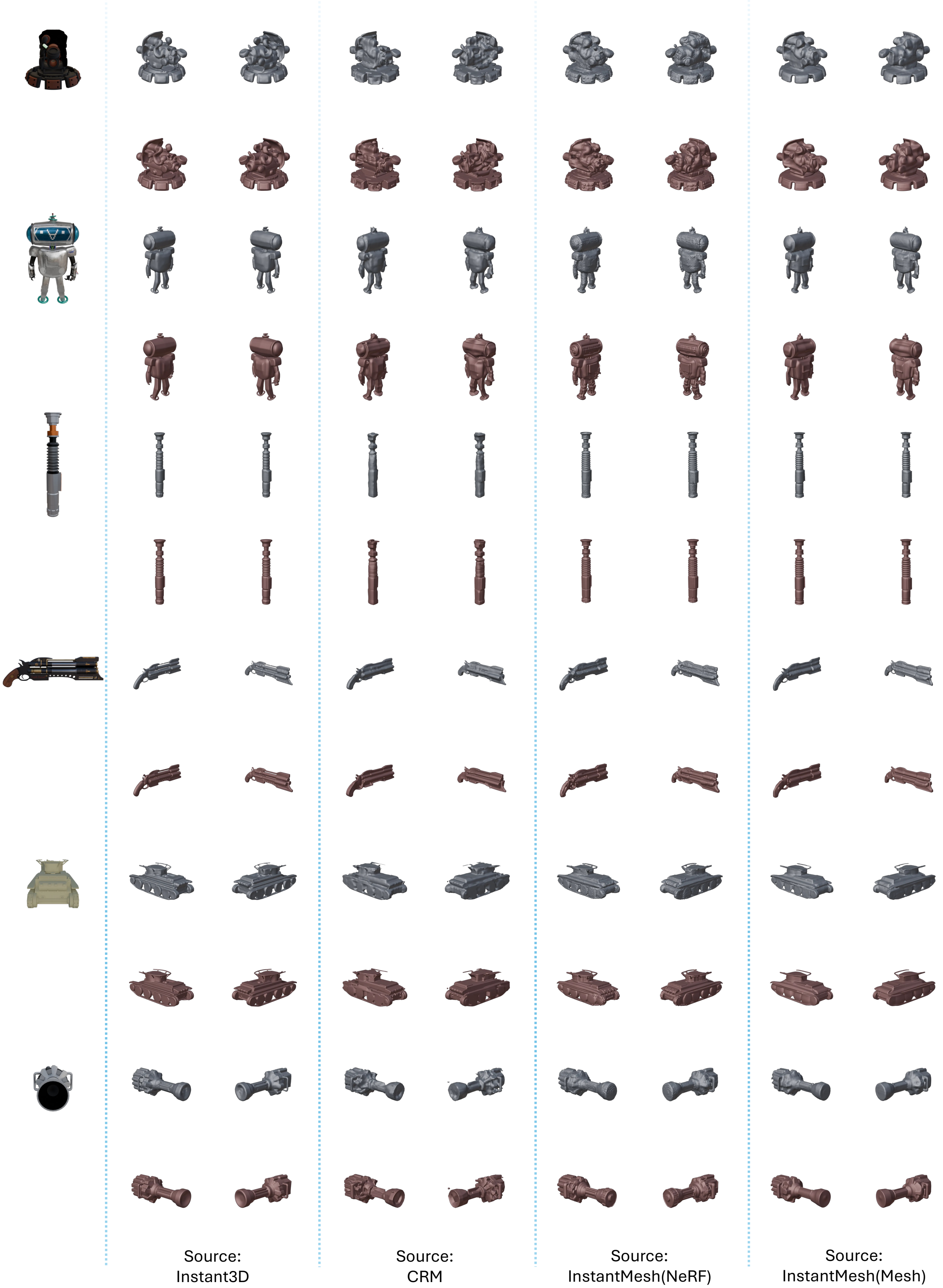

Figure 12. Feed-forward reconstruction results on Objaverse. 🟦 represent coarse, 🟥 represent fine refinement results from our method.

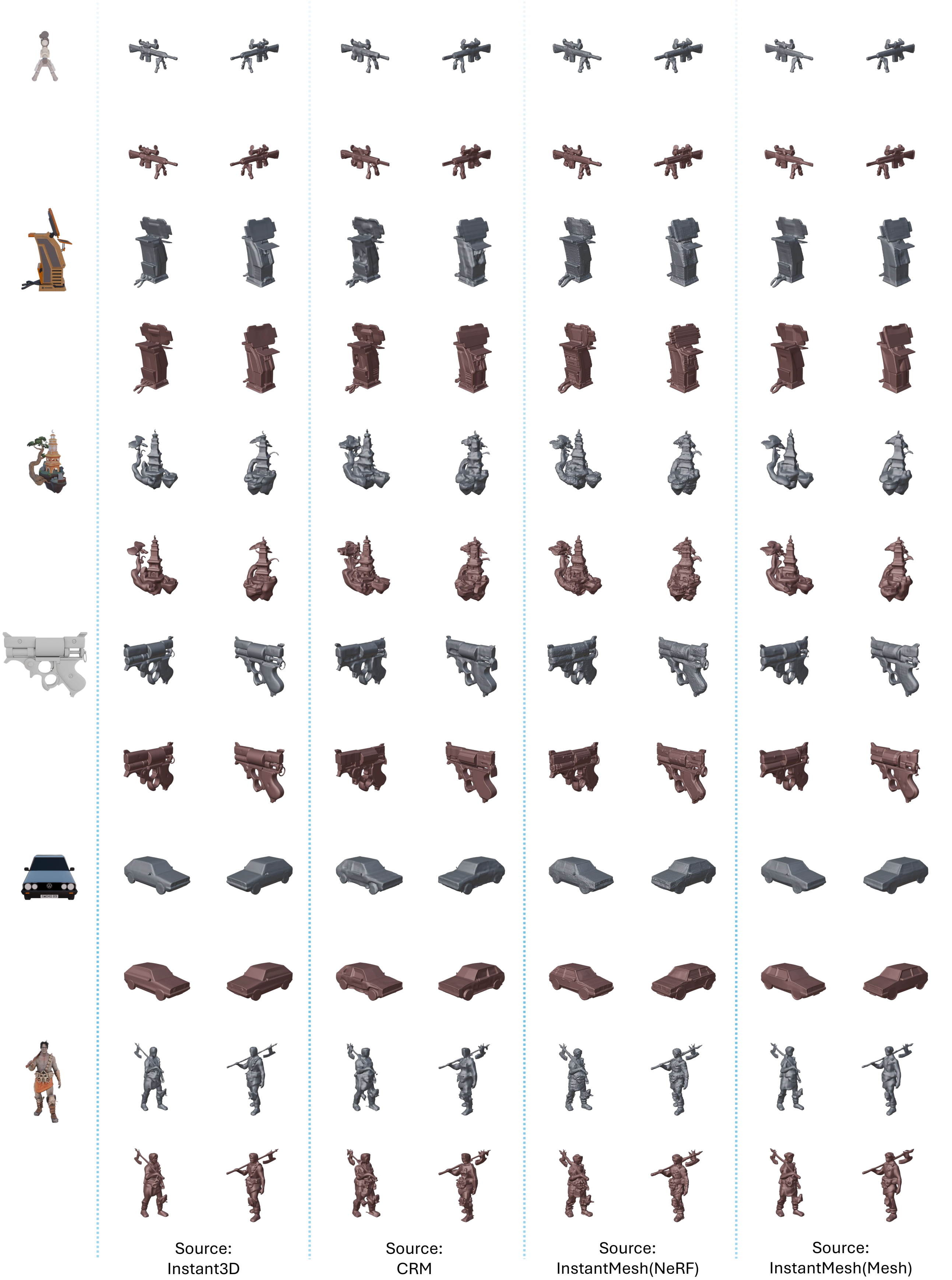

Figure 13. Feed-forward reconstruction results on Objaverse. ■ represent coarse, 🟥 represent fine refinement results from our method.

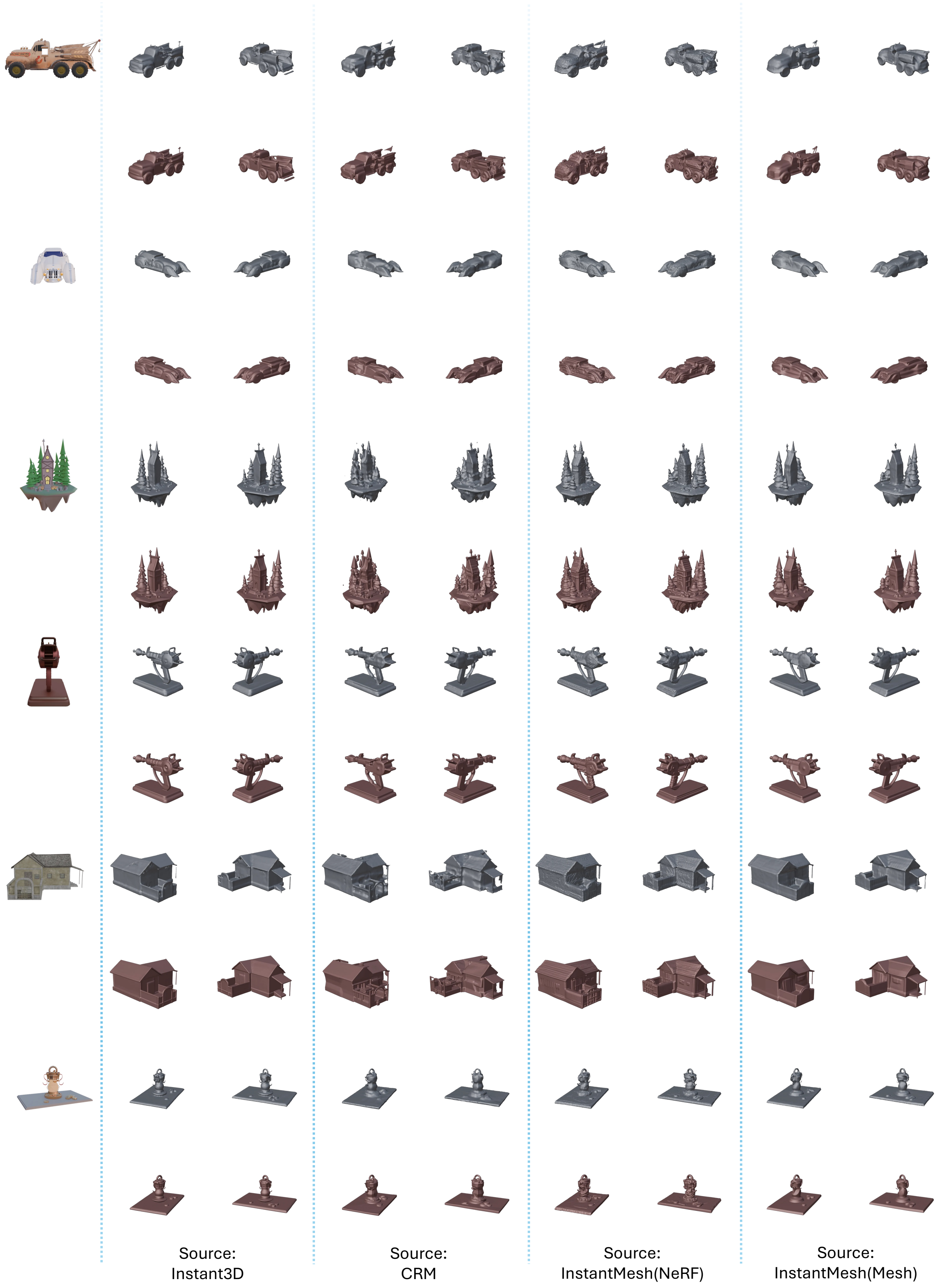

Figure 14. Feed-forward reconstruction results on Objaverse. ■ represent coarse, 🟥 represent fine refinement results from our method.

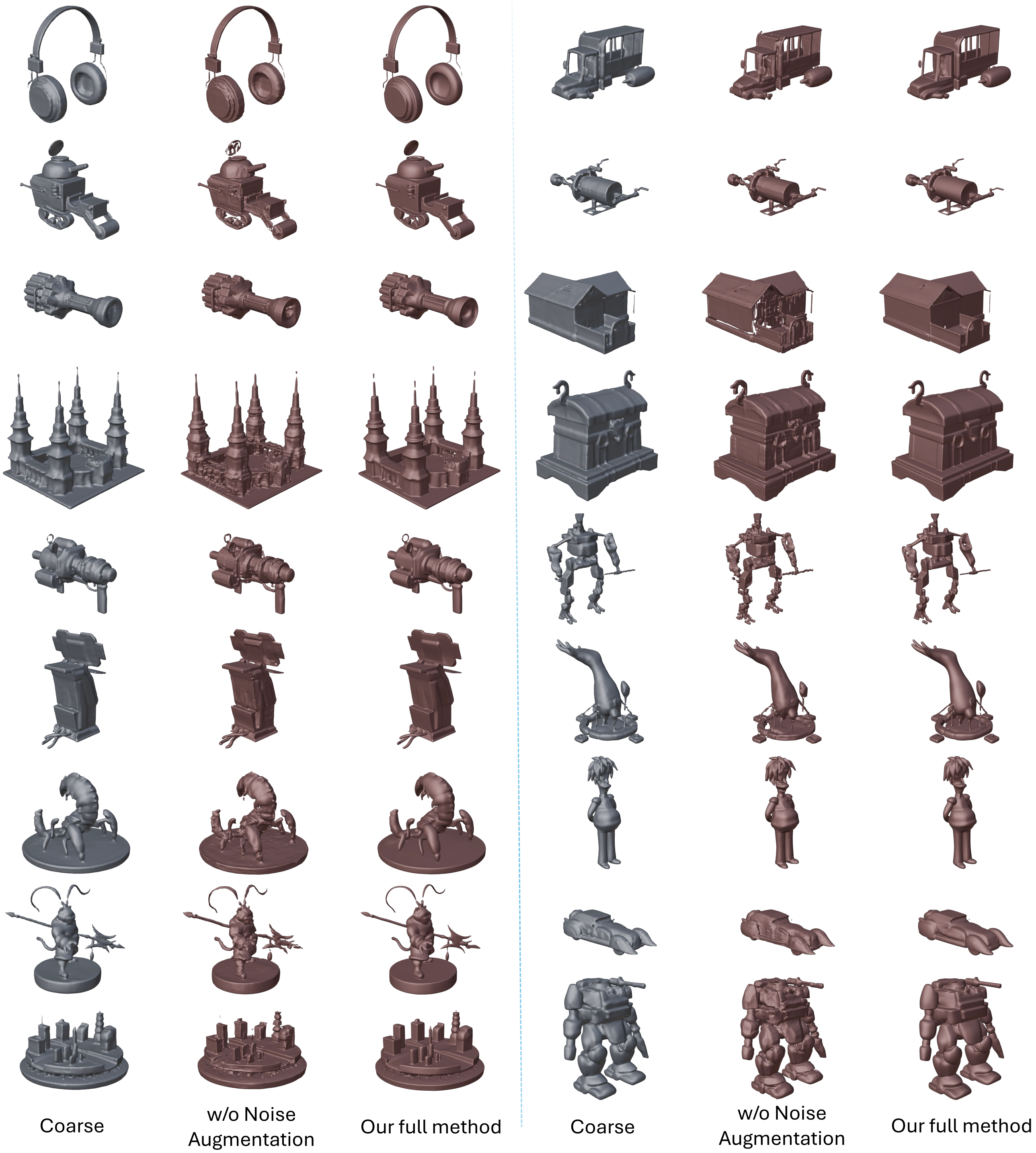

Figure 15. Ablation study visual results.

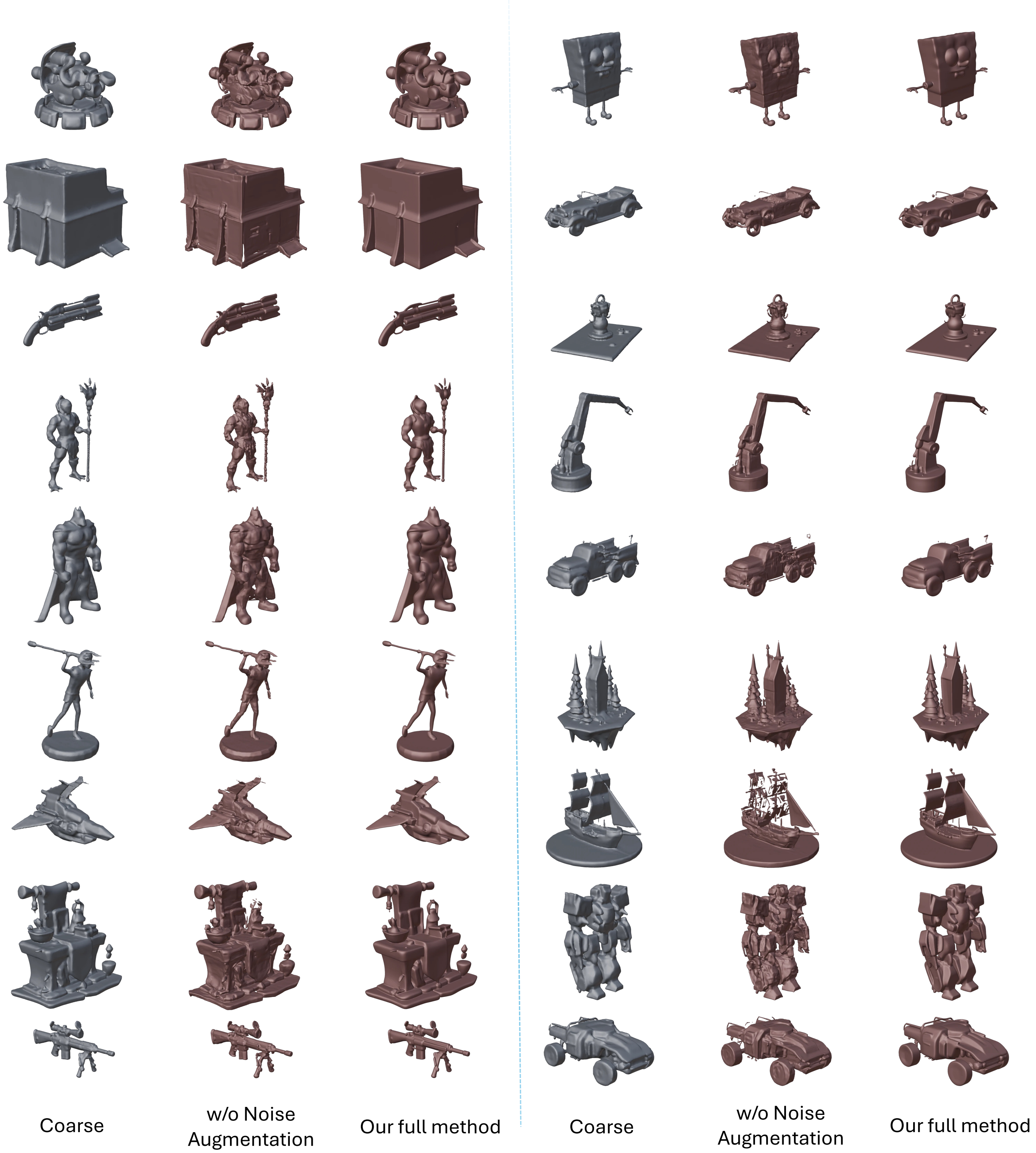

Figure 16. Ablation study visual results.