Method	GSO GD↓	OmniObject GD↓
Orient Anything [39]	27.7	34.1
Reflect3D-FF (Ours)	<u>22.7</u>	<u>31.1</u>
Reflect3D (Ours)	**13.3**	**22.8**

Table 5. Quantitative comparison of our symmetry detection method against Orient Anything [39].

Method	CLIP-Sim↑
DreamGaussian [36]	0.803
Ours	**0.838**

Table 6. CLIP similarity of our symmetry-conditioned 3D generation method, evaluated on a subset of object with asymmetric details selected from the GSO dataset. Our method still brings improvements even though the objects are not perfectly symmetric.

Method	CLIP-Sim↑
Ours	0.768
w/o Symmetry Alignment	0.644
w/o Symmetric SDS	0.738
w/o Symmetric Densification	0.717
w/o Symmetric Texture Refinement	0.731

Table 7. Quantitative ablation studies for our symmetry-conditioned 3D generation method on a subset of the GSO dataset. Removing each of our components degrades the performance.

9. Appendix

In this appendix, we present 1) a detailed description of our ground truth symmetry plane generation method; 2) more implementation details for our Reflect3D symmetry detector and our single-image 3D generation pipeline. All references and citations in this supplementary document refer to the main paper.

9.1. Dataset

In this section, we provide more details on generating ground truth reflection symmetry for our training and evaluation datasets. This generation method is also used in our *Shape to Symmetry* baseline method.

Automatic Ground Truth Generation. We propose a reliable protocol to efficiently generate accurate symmetry ground truth for arbitrary 3D datasets. First, for each mesh in the 3D dataset, we center it by its bounding sphere center and normalize it by scaling its bounding sphere to a unit sphere. This ensures the center of any potential ground truth symmetry plane is on the origin. Then we uniformly sample N_p points from this mesh. To find the symmetry plane, we generate a set of N_c unit vectors uniformly spanning a unit hemisphere as candidates for symmetry plane normals, analogous to the symmetry hypothesis in our Reflect3D single-image symmetry detector. These candidates only span a hemisphere rather than the entire unit sphere because a normal vector $\mathbf{n}$ and its opposite $-\mathbf{n}$ represent the same plane. For each candidate normal, we derive its corresponding candidate plane as the plane passing through the origin and having a normal as the candidate normal. We choose a large enough number of sampled points N_p and symmetry candidates N_c, empirically we use $N_p = 50000$ and $N_c = 31$.

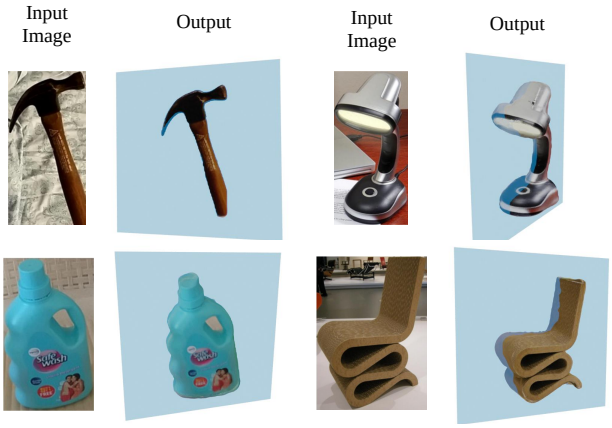

Figure 7. Additional symmetry and 3D generation results on in-the-wild images from [6].

We reflect the point cloud by each candidate plane. Then by how well the reflected point cloud and the original point cloud align, we can infer whether the plane is a ground truth symmetry plane of the shape. We calculate a Chamfer distance between the original and reflected point clouds. We manually select a threshold to eliminate low-quality symmetry planes. We determine this threshold after comprehensively examining the results and making sure it aligns with human perception of symmetry. Meanwhile, for candidate planes that pass the threshold, we can refine them by registering the reflected point cloud to the original point cloud. We apply the iterative closest point (ICP) algorithm for this registration. Then we derive a refined symmetry plane from the registered point clouds.

9.2. Implementation Details

Reflect3D. Our method employs a frozen DINOv2 backbone with the ViT-L/14 architecture. The symmetry decoder consists of 4 layers with 256 channels. Our Adam optimizer uses a weight decay of 0.05. The classification loss and regression loss are weighted at 1.0 and 0.1, respectively. Training is performed on 2 A100 GPUs for 2 days. For multi-view enhancement, we generate 8 views with the same elevation as the input view, uniformly spacing them in azimuth such that adjacent views are $45°$ apart. Finally, a clustering threshold of $30°$ is applied in the aggregation step.

Single-Image 3D Generation. During symmetry alignment, we disable the image-based MSE loss and apply only the symmetrically sampled SDS loss based on the detected

Figure 8. Symmetry and 3D generation result of our method combined with SPAR3D [13] on ImageNet images.

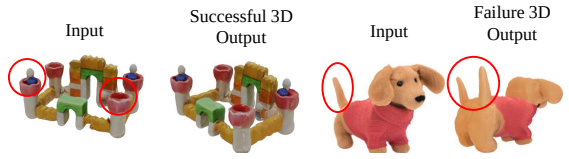

Figure 9. Failure cases on objects with minor asymmetry.

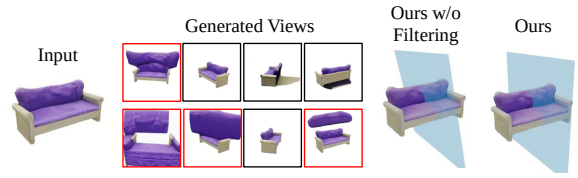

Figure 10. Illustration of inconsistent views filtering and corresponding failure cases.

symmetry. We run this optimization stage for 400 steps to obtain more accurate symmetry planes, though fewer steps often suffice in practice. We convert the rough Gaussian splatting to a mesh using the marching cubes algorithm. We remove internal surfaces via ambient occlusion, and uniformly sample 10,000 points from the mesh surface to align the symmetry plane, as discussed in Section 3.4. In the subsequent symmetric SDS optimization stage, we optimize for 500 steps following DreamGaussian [36]. Finally, we refine the texture for 50 steps.

9.3. Additional Results

Additional in-the-wild results. We show additional results on in-the-wild images taken from [6] in Figure 7. Our method robustly estimate symmetry and generate 3D textured shapes from these images.

Integration with SPAR3D [13]. We integrate our approach with a SOTA image-to-3D method, SPAR3D [13], and show our symmetry and 3D results on ImageNet images in Figure 8. SPAR3D uses a diffusion model to generate a sparse point cloud from an input image and reconstructs a 3D mesh conditioned on the points. We use our Reflect3D to detect symmetry and align the symmetry to the SPAR3D point cloud through our *symmetry alignment*, then we perform *symmetric densification* (Sec 3.4). The symmetrized point cloud then guides the mesh generation in SPAR3D. We show our Reflect3D robustly generalizes to in-the-wild ImageNet images. In addition, integrating Reflect3D into SPAR3D leads to significantly better reconstructions on symmetric objects. Our method corrects symmetry errors, improves symmetric details, and reduces the perspective-induced skew in SPAR3D generation. This validates the generality of symmetry prior to different 3D generation methods.

Comparison to Orient Anything [39]. Orient Anything (OA) [39] is a recent work that estimates object orientation. Estimating object orientation is not equivalent to detecting symmetry, resulting in several failure cases: 1) Most objects exhibit left-to-right symmetry, but some—like mugs and hairdryers—have front-to-back symmetry, which OA can not distinguish. 2) Some objects, like tables, have meaningful symmetries but ambiguous canonical poses, causing OA to return null results. We compare Reflect3D with OA using left-to-right symmetric convention in Table 5, and our method achieves better results on GSO and OmniObject3D. We follow OA's practice in data curation and take the XZ plane relative to the canonical pose (left-to-right symmetry from the canonical front view) as the symmetry plane. Our feedforward model outperforms OA on both datasets.

Objects with minor asymmetry. In Table 6, we present 3D generation performance, measured by CLIP similarity, on a subset of objects with minor asymmetry from GSO. Our method still benefits the 3D generation of objects with minor asymmetry. Meanwhile, we show the failure cases on these objects in Figure 9. When an object part and its asymmetry-counterpart are both visible in the input image, our method typically respects the asymmetry. But when only one half is visible, the problem becomes highly ill-posed and our method may incorrectly predict symmetry and make mistakes in reconstruction.

Filtering inconsistent views in Reflect3D. Inconsistent views add noise to the symmetry prediction. To remove them, we compute CLIP similarity between each generated view and the input image, and drop the low-score views.

The threshold is a hyperparameter tuned to maximize alignment to human annotations of view consistency on Objaverse validation set. We further annotated the view consistency of 180 generated views on GSO to test the robustness of our filtering method. Our results align with human annotation on 93.9% of sampled views. We include failure cases caused by not removing inconsistent views in Figure 10.

Quantitative ablation studies of 3D generation. We present a quantitative ablation study of our 3D generation method in Table 7. Each component of our method is crucial and removing each one causes a performance drop.