

# REVIVE 3D: Refinement via Encoded Voluminous Inflated prior for Volume Enhancement

(Supplementary Material)

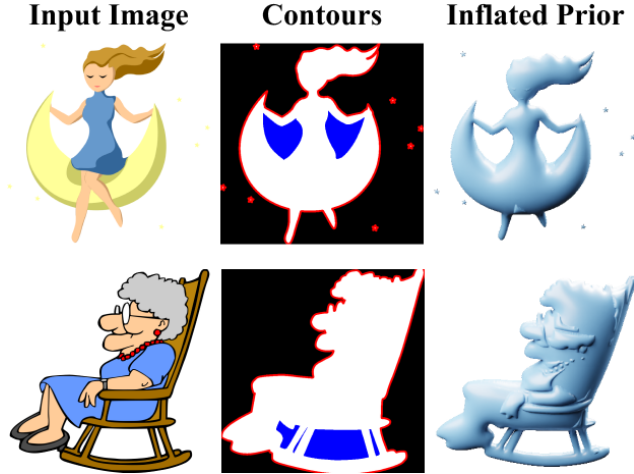


Figure 11. Contour and Cut Visualization. The outer boundary is marked in red, and the excluded internal regions are filled in blue.

## A. More Implementation Details

### A.1. Generation of the Inflated Prior

We automatically generate an Inflated Prior to serve as a volumetric and part-aware prior for Stage 2. This is done by combining a global silhouette mask with local part masks derived from an automatic segmentation method.

The silhouette mask is a binary mask generated from the input image, designating the foreground object area as 255 and the transparent background as 0. The boundaries of the foreground regions define one or more outer contours. Simply inflating these outer contours as a single solid would cause interior empty regions to be filled and disconnected parts to be merged, leading to a loss of the original overall shape. To prevent this, we explicitly exclude these internal empty areas from the inflation process (Fig. 11).

To obtain part-aware local regions, one could in principle apply traditional edge detection. However, standard edge detectors struggle to accurately capture subtle boundaries and often fail to produce precise, closed contours that are usable for our inflation process. Therefore, instead of edges, we rely on an automatic segmentation-based approach [36] to extract local parts from the input image. The resulting segmentation masks vary greatly in size, location, and shape, so we apply a filtering process based on four specific criteria (Fig. 12).

First, we filter based on **Margin**. We only keep masks that lie completely inside the main silhouette and discard any mask that extends outside the silhouette boundary. Masks that lie too close to the silhouette contour are also problematic, because the height field is constrained to zero along the contour. Inflating such masks can introduce rendering artifacts or distort the final outline. To avoid this, we remove any mask judged to be too close to a boundary. Concretely, a mask is removed if it is located within 6 pixels of the main silhouette contour, if it overlaps with the 3-pixel eroded region of the contour, or if it is within 16 pixels of an internal empty region. Second, we eliminate **Overlap**. We sort all masks by size in descending order to establish priority. Any smaller mask that overlaps with an already selected larger mask is discarded. While smaller masks can represent fine details, we prioritize the larger, more dominant cues for providing clear guidance in Stage 2. Third, we apply a **Size** constraint. Masks smaller than 16 pixels are removed as they are typically noise or are difficult to sample. Conversely, masks larger than 10,000 pixels are also removed because they can overwhelm and obscure other useful detail cues. Finally, we correct for **Irregularity**. Unusually thin masks with an aspect ratio exceeding 1:5 are removed, as such thread-like regions tend not to be refined and often appear as distracting artifacts in the final mesh. We also use a 3-pixel erosion test to assess the mask’s thickness. Line-like masks that lose over 95 percent of their original area upon erosion are discarded. Masks that lose between 60 and 95 percent of their area are considered viable but thin, so they are reinforced with a 5-pixel dilation and then used.

The resulting set of filtered masks is then superimposed onto the global silhouette. This process creates the final Inflated Prior, which provides the necessary detailed part cues for Stage 2.

### A.2. Analysis of Inflated Prior

**Effect of Prior Specificity.** To demonstrate the importance of the volumetric and part-aware cues provided by the proposed Inflated Prior when generating 3D meshes from flat images, we perform a deformation analysis. Stage 1 of our framework uses the input image silhouette and segmentation masks to construct an Inflated Prior that captures image-specific volume and local details. If this image-specific prior is replaced by a generic 3D mesh (Fig. 13), the

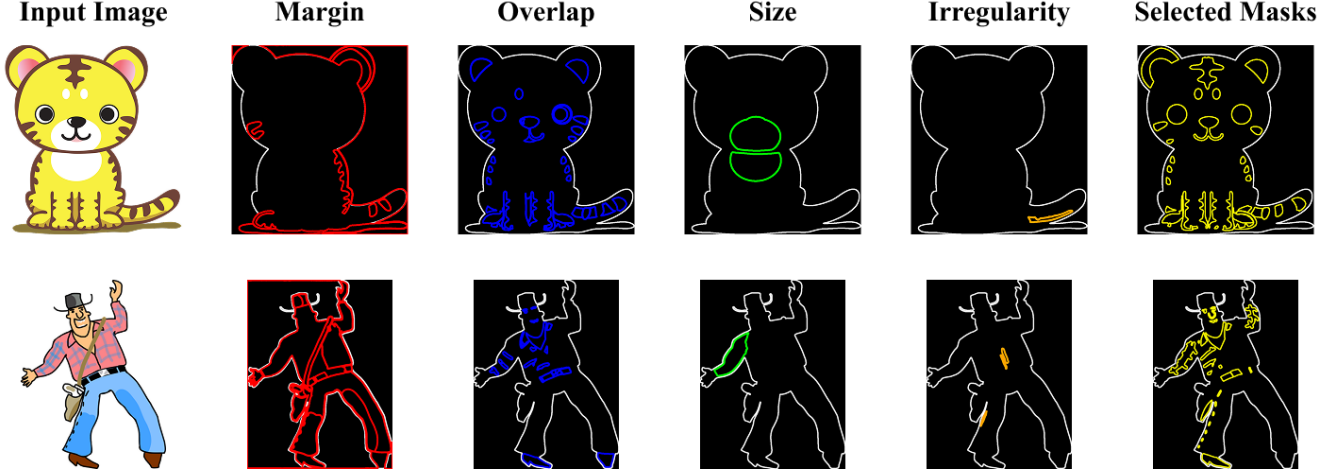


Figure 12. Visualization of the mask filtering process. We visualize masks filtered by our four criteria: Margin (red), Overlap (blue), Size (green), and Irregularity (orange). The final column shows the aggregated selected masks (yellow) that pass all filters.

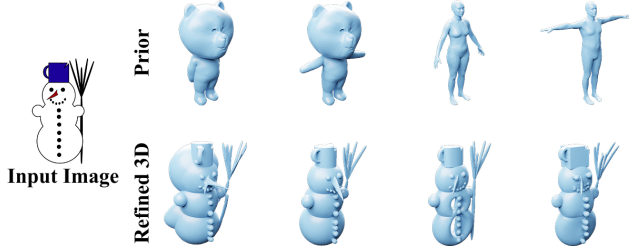


Figure 13. Prior and refined 3D results when using character and human body template meshes as priors.

subsequent 3D Latent Refinement stage cannot fully reflect the structures in the input image and instead produces results that rely on the generic prior shape.

To quantify how this prior specificity influences the final output, we introduce an interpolation weight called *Sphericity weight* (Fig. 14). Sphericity weight linearly interpolates vertex positions between the original Inflated Prior and a sphere, and therefore controls how close the prior shape is to the original image-specific Inflated Prior or to a sphere. When Sphericity weight is 0.0, the vertex coordinates of the original Inflated Prior are used without modification, which corresponds to the most image-specific setting. When Sphericity weight is 1.0, all vertices are normalized so that they lie at an equal distance from the center, resulting in a perfect sphere. As Sphericity weight increases from 0.0 to 1.0, the prior gradually loses image-specific cues and converges toward the shape of a sphere. This analysis confirms that constructing the Inflated Prior in an image-specific form plays a crucial role in the quality of the final 3D generation.

**Effect of Inflation Strength  $c$ .** We also analyze how the inflation strength value  $c$ , used when inflating the silhouette and the filtered masks, affects both the initial Inflated Prior and the final generated 3D mesh. In our framework, the global inflation strength applied to the silhouette and the local inflation strength applied to the segmentation masks can be controlled independently. Using the default setting  $c$  equal to 1.5 as a reference, we conduct comparison experiments by increasing each of the global and local values to 3 and 6, which correspond to two and four times the default strength.

The results (Fig. 15) show that increasing the global inflation strength causes large-scale structures to become overly inflated, which distorts and eventually breaks the overall silhouette. In contrast, increasing the local inflation strength exaggerates fine-scale features, leading to results where small details are emphasized. Based on these observations, we set the default inflation strength to  $c$  equal to 1.5 for both global and local inflation in our experiments.

### A.3. Compactness and Normal Anisotropy

**Mesh preprocessing.** Before computing the metrics, we normalize each input mesh. If a file contains a scene with multiple mesh instances, we first merge all geometry into a single triangular mesh and remove unreferenced vertices and degenerate faces. We then translate the mesh so that its centroid is at the origin and rescale it so that the longest side of its oriented bounding box has unit length. This preprocessing makes Compactness and Normal Anisotropy invariant to global translation and scale and reduces numerical issues caused by extreme coordinate ranges.

**Compactness.** For Compactness we require reliable estimates of the enclosed volume  $V$  and surface area  $S$  even

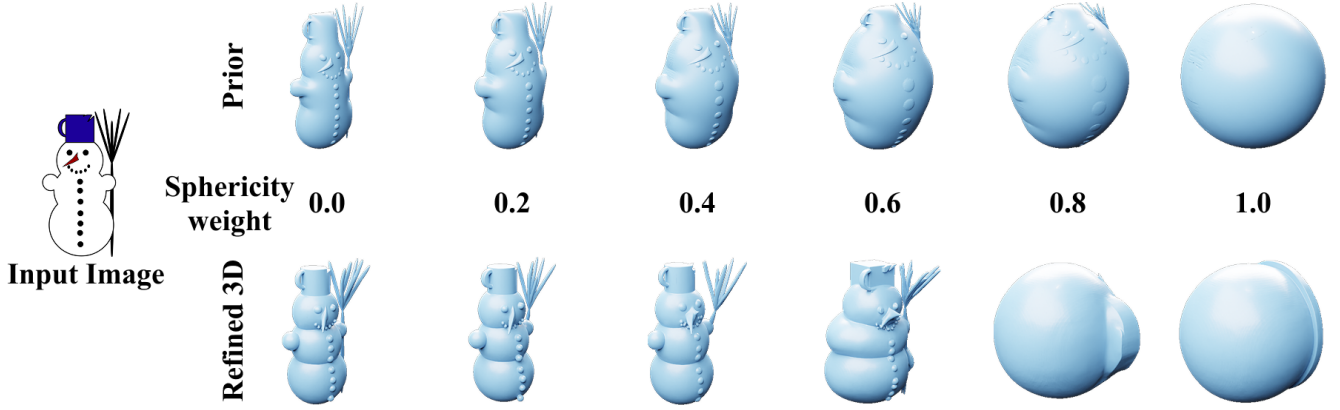


Figure 14. Interpolating the Inflated Prior toward a sphere degrades image-consistent geometry, as shown by priors (top) and refined 3D results (bottom) for different Sphericity weights from 0.0 to 1.0.

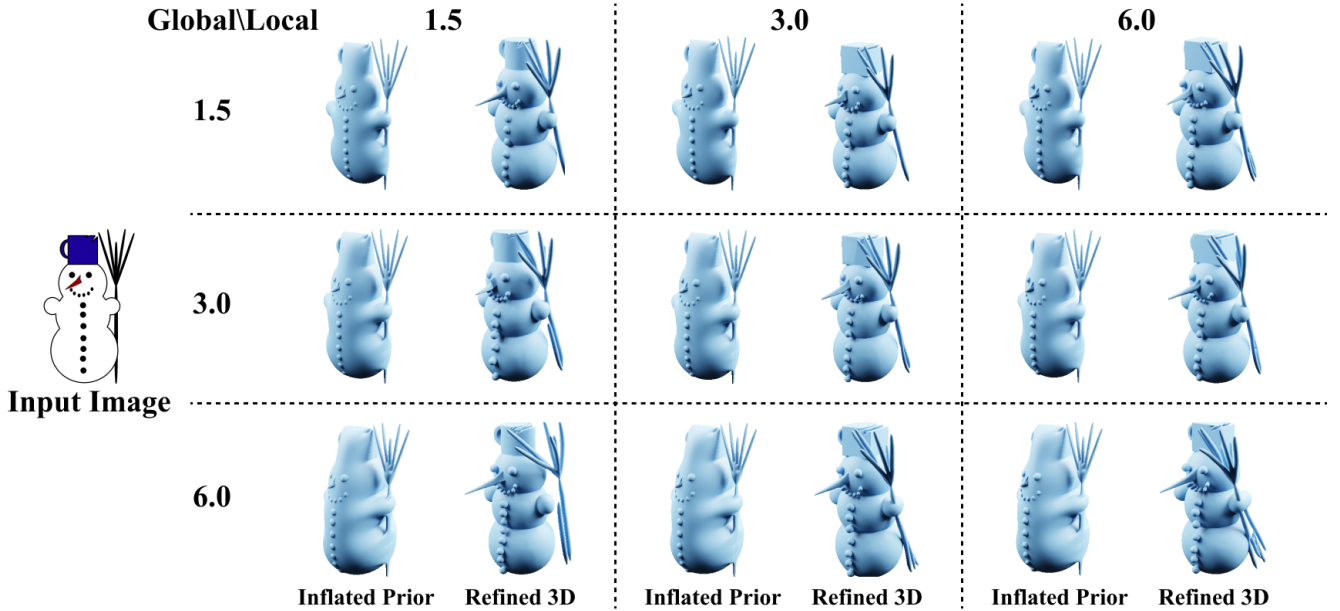


Figure 15. Effect of varying global and local inflation strength (1.5, 3.0, 6.0) on the Inflated Prior and refined 3D results.

for imperfect meshes. If a mesh is reported as watertight by `trimesh`, we first query the analytic volume and area returned by the library. When both values are finite and strictly positive we directly use them as  $V$  and  $S$ .

For non-watertight or numerically unstable meshes we instead use a voxel-based estimate. We voxelize the normalized mesh with a voxel pitch of 0.02 in all our experiments and obtain a binary occupancy grid. We then regularize this grid using simple volumetric morphology to approximate a single watertight solid and suppress small artifacts. We fill internal cavities by three-dimensional hole filling and apply morphological closing with a spherical structuring element so that thin gaps or cracks in the surface are bridged. We then keep only the single largest six-connected component

of occupied voxels and discard small floating pieces, which would otherwise add unstable contributions to the volume and surface area. The volume is computed by counting occupied voxels and multiplying this count by the voxel volume, and the surface area is computed by extracting an isosurface with marching cubes and measuring the triangle area of the resulting mesh. When marching cubes fails we instead count all voxel faces that lie on the boundary between occupied and empty cells.

Finally we compute Compactness from  $V$  and  $S$  using the definition in Eq. 5 and clamp the result to the range  $[0, 1]$ . If either  $V$  or  $S$  is non-positive we set  $C = 0$ .



Figure 16. Category-wise Compactness (C) and Normal Anisotropy (NA) on ModelNet40 [53].

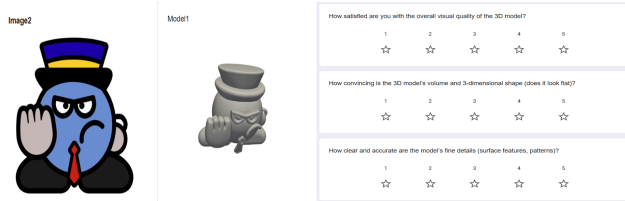


Figure 17. Example of the web interface used in our user study.

**Normal Anisotropy.** Normal Anisotropy is computed from the distribution of per-face normals, weighted by face area. For each mesh we take the face normals and the corresponding face areas, normalize the normals to unit length, and assign each face an area weight obtained by dividing its area by the total surface area. We then map every normal to spherical coordinates on the unit sphere using its azimuth angle and its vertical component. This choice makes a uniform grid in this two-dimensional space correspond to patches of roughly equal area on the sphere.

In this two-dimensional space we build a histogram with  $K$  bins in total, with  $K = 128$  in all our experiments. We choose the number of azimuth and elevation bins so that their product equals  $K$ , and accumulate the area weights of

faces whose normals fall into each bin. The resulting bin values are renormalized to sum to one and define a discrete probability distribution over normal directions. We compute the Shannon entropy of this distribution and normalize it by the entropy of a uniform distribution with  $K$  bins. Normal Anisotropy is defined as one minus this normalized entropy as in Eq. 6, which by construction lies in the range  $[0, 1]$ .

**ModelNet40 category-wise statistics.** We evaluate Compactness and Normal Anisotropy for all categories in the ModelNet40 [53] dataset and report the per-category statistics in Fig. 16. Meshes for which loading or metric computation fails are excluded from the averages.

#### A.4. User study interface

We conducted the user study using a web-based interface (Fig. 17). For each trial, participants were shown the 2D input image together with an auto-rotating GIF (360-degree render) of one of the 3D models, and were asked to answer the three questions described in the main paper using a 5-point Likert scale. Across 51 participants and 6 methods, this resulted in a total of 918 individual ratings.





Figure 18. Dataset examples.

### A.5. Dataset

We collected 2,232 flat images with limited 3D cues and show a few representative examples in Fig. 18.

### B. Image-Conditioned 3D Editing

**Editing comparison.** Our pipeline’s latent refinement process can be naturally repurposed for 3D editing. Instead of using the Inflated Prior, we encode a source model to obtain the initial latent  $z_0$  and provide an edited image ( $y_{\text{edited}}$ ) as the condition. We then add Gaussian noise according to Eq. 4 at the initial noise level  $t_0$ , reducing fidelity to the source while preserving its coarse structure. Subsequently, denoising under the new edited condition  $\psi(y_{\text{edited}})$  steers the latent toward the desired edits. This approach enables high-quality, image-conditioned 3D editing without requiring an explicit 3D editing mask. We demonstrate this application using the Edit3DBench dataset [20], with qualitative results shown in Fig. 19.

In addition, we show qualitative comparisons in Fig. 20. For these experiments, we use Hunyuan3D-2.1 [45] as the 3D latent backbone and compare our editing results against

single image-to-3D generations obtained from the same backbone. When editing from a source model, the original geometry is retained in the initial latent representation, so our method preserves the structural details of the source mesh much better than direct generation. For example, the curvature of a tail, wrinkles on the hands, the height of a basket, and wrinkles in clothing remain clearly visible after editing. These results indicate that our approach can modify a 3D mesh to match an edited image while faithfully preserving the source structure.

### C. More Results

#### C.1. Effect of the backbone.

The behavior of Stage 2 (3D Latent Refinement) depends on the choice of the pretrained conditional 3D latent generative backbone (Fig. 21). When Hunyuan3D-2.1 [45] is used as the backbone, the model has been trained to produce rich surface appearance and fine-scale details, so our refinement yields volumetric meshes that preserve these detailed surface structures. In contrast, Direct3D [52] is more specialized for recovering the overall shape rather than fine-

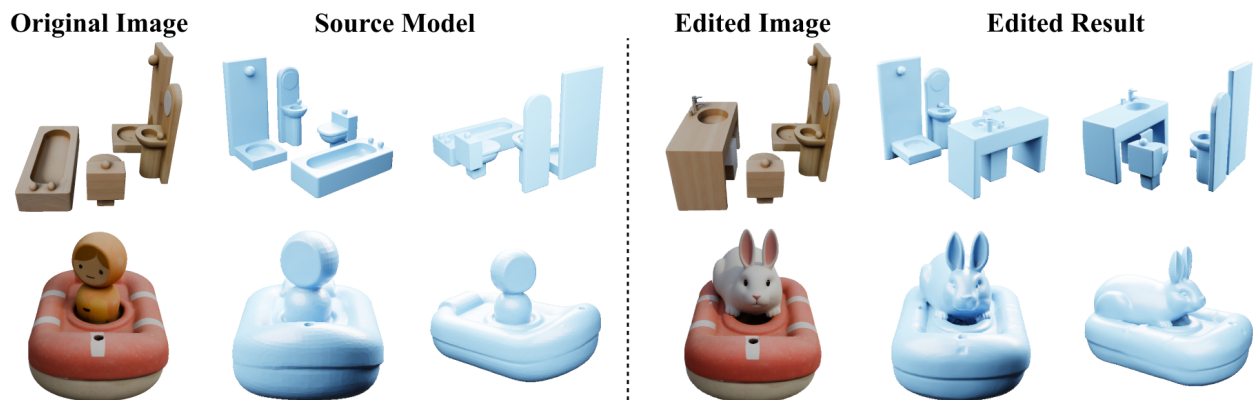


Figure 19. Image-conditioned 3D editing results on the Edit3DBench dataset [20].

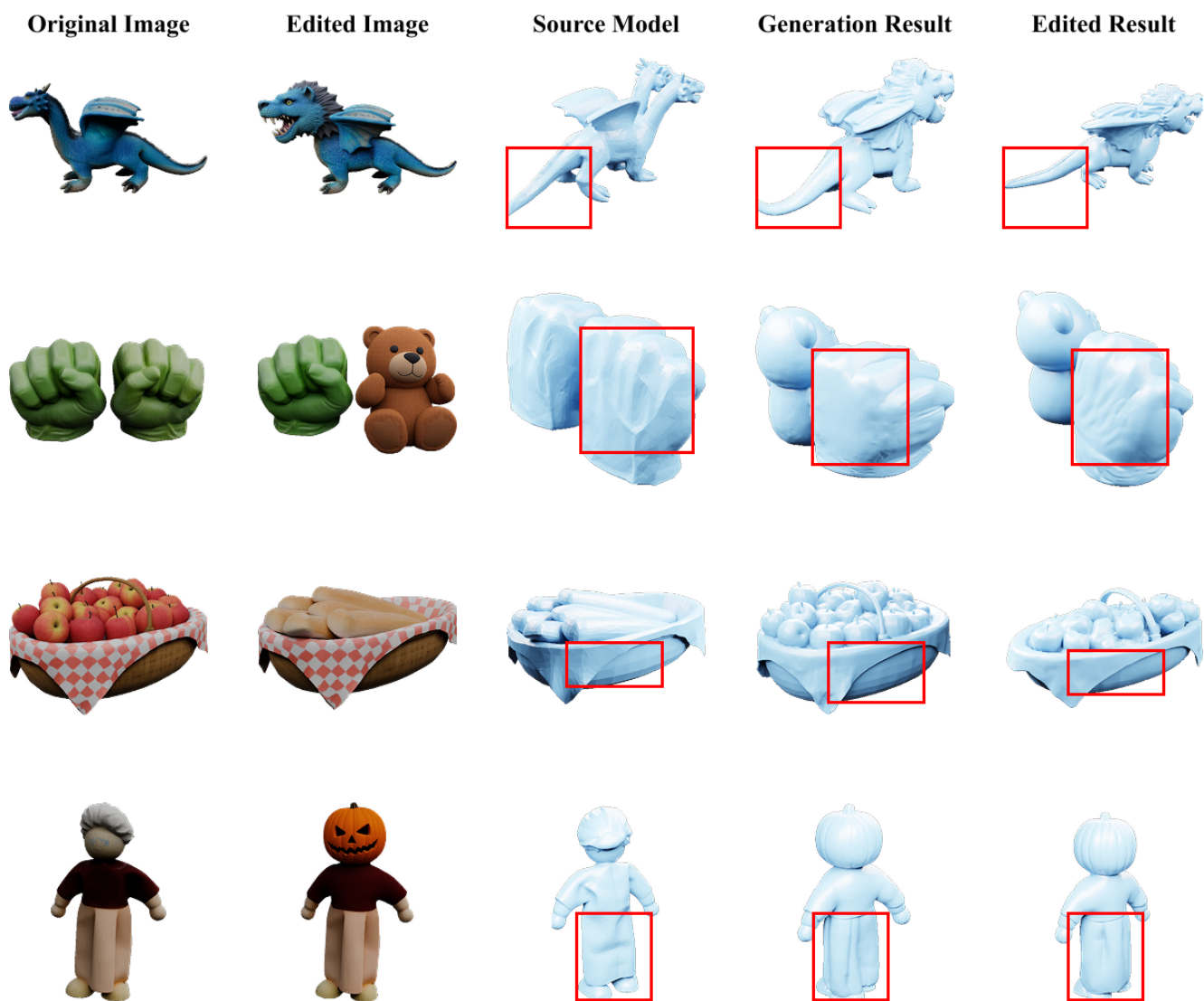


Figure 20. Image-conditioned 3D editing result on the Edit3DBench dataset [20]. Generation result denotes the single image-to-3D reconstruction, and the red boxes highlight regions used to check how well the original source geometry is preserved.

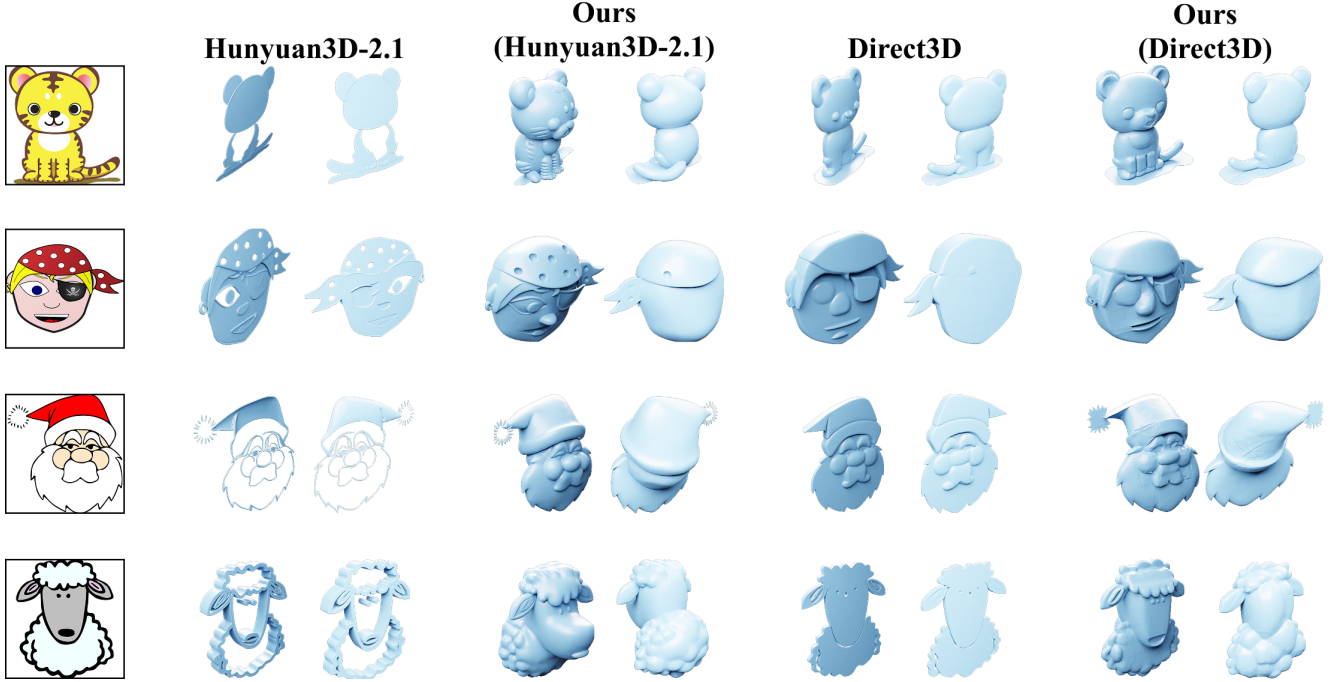


Figure 21. Effect of different 3D latent backbones.

grained surface structure, and under the same framework it tends to emphasize the global geometry while providing relatively weaker local details. These comparisons indicate that our method consistently produces more volumetric 3D meshes across different backbones, while effectively leveraging the pretrained 3D knowledge encoded in each backbone.

## C.2. More Comparisons

We conduct an additional experiment that evaluates 3D reconstruction from flat images, and show representative qualitative results in Fig. 22. For all baseline methods, we follow their official configurations as closely as possible to ensure a fair comparison.

For DrawingSpinUp [69], we observed that the default setting sometimes applies overly aggressive thinning on flat inputs, which can prevent any 3D mesh from being generated. In such failure cases, we rerun the method with the thinning step disabled and use these results instead.

Hunyuan3D-Omni [46] requires a 3D bounding box as a conditioning input. To obtain the best possible performance, we set the  $x$  and  $y$  extents from the normalized foreground bounding box of the input image and fix the depth extent  $z$  to 0.6 in all our experiments.

**Texture comparison.** Trellis [54] directly generates textured 3D meshes, whereas all other methods produce geometry-only meshes. To compare textured appearance,

we apply the Hunyuan3D-Paint pipeline from Hunyuan3D-2.1 [45] to each geometry-only mesh, using the original 2D image as the conditioning input for texture synthesis. The resulting textured meshes are shown in Fig. 23.

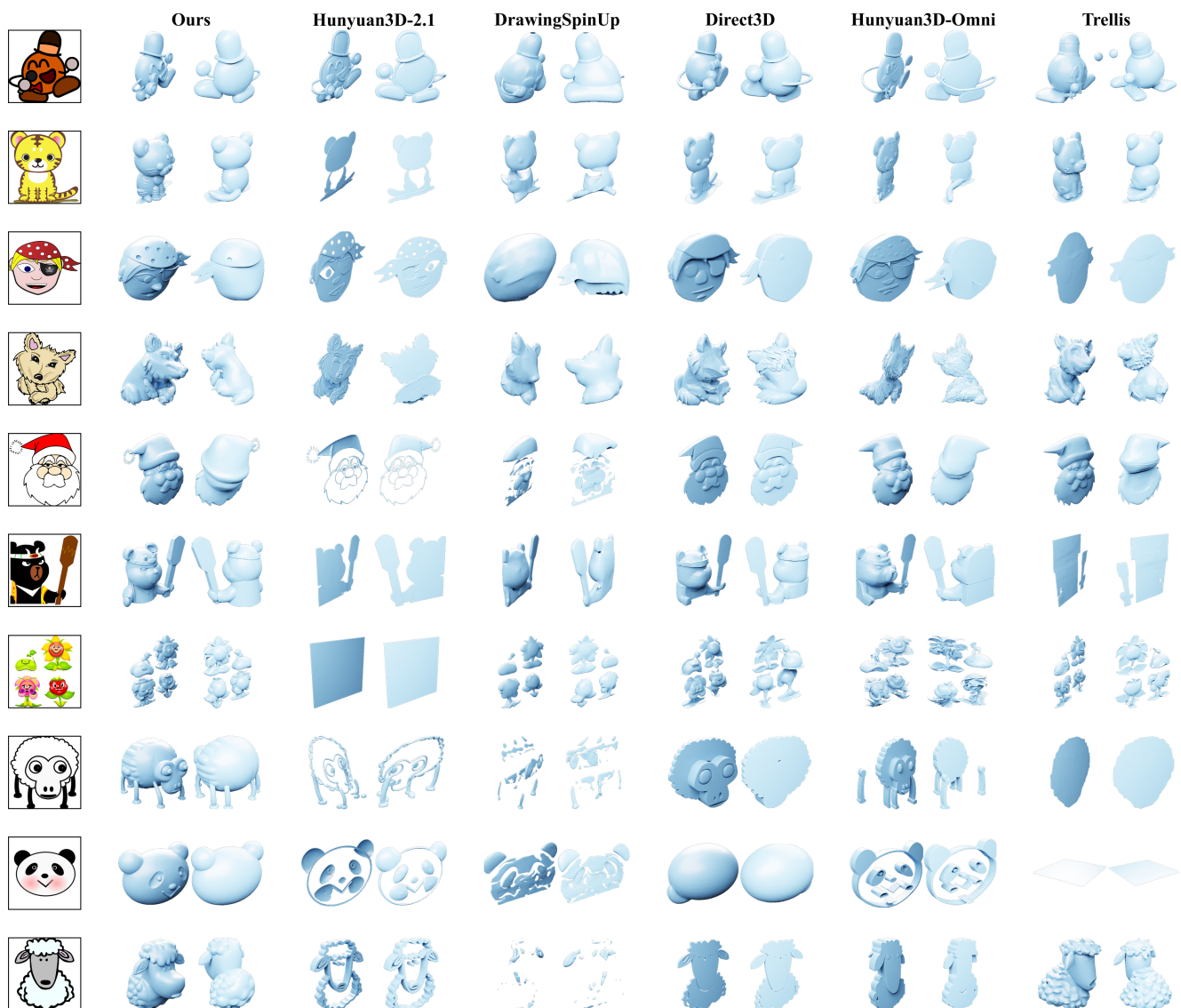


Figure 22. Additional qualitative comparisons of 3D meshes generated from flat images by our method and prior works.





Figure 23. Texture comparison of 3D generations from flat images.