Acknowledgements

We thank Ellis Brown, Fred Lu, Sanghyun Woo, Adithya Iyer and Oscar Michel for helpful discussions. This work was supported in part through the NYU IT High Performance Computing resources, services, and staff expertise. This research is also supported by Intel, Google TRC program, the Google Cloud Research Credits program with the award GCP19980904, and an Amazon Research Award Fall 2023.

A. Generative Enhancement Details

Algorithm 1: Generative Enhancement

Define: Pre-trained 2D text-to-image diffusion model M, input image I, coarse image I_c, enhanced image I_f, inversion prompt y_{inv}, prompt y, depth map D

$\hat{M} \leftarrow$ FINE-TUNE DREAMBOOTH(I)
$\boldsymbol{x}_0^c, \dots \boldsymbol{x}_T^c \leftarrow$ DDIM-INVERSION($I_c, y_{inv}, D; \hat{M}$);
$\boldsymbol{x}_T^f \leftarrow \boldsymbol{x}_T^c$
for $t \leftarrow T$ **to** 0 **do**
  $\boldsymbol{f}_t^c, \boldsymbol{A}_t^c \leftarrow \epsilon_\theta(\boldsymbol{x}_t^c, y_{inv}, D, t; \hat{M})$
  $\boldsymbol{f}_t^f, \boldsymbol{A}_t^f \leftarrow \epsilon_\theta(\boldsymbol{x}_t^f, y, D, t; \hat{M})$
  if $t > \tau_f$ **then** $\boldsymbol{f}_t^f \leftarrow \boldsymbol{f}_t^c$
  if $t > \tau_A$ **then** $\boldsymbol{A}_t^f \leftarrow \boldsymbol{A}_t^c$
  $\boldsymbol{\epsilon}_{t-1}^f \leftarrow \epsilon_\theta(\boldsymbol{x}_t^f, y, D, t; \boldsymbol{f}_t^f, \boldsymbol{A}_t^f; \hat{M})$
  $\boldsymbol{x}_{t-1}^f \leftarrow$ DDIM-DENOISING($\boldsymbol{x}_t^f, \boldsymbol{\epsilon}_{t-1}^f; \hat{M}$)
end
Result: $I_f \leftarrow$ DECODER($\boldsymbol{x}_0^f$)

The generative enhancement pipeline starts with fine-tuning DreamBooth [67] with the input image. Subsequently, we apply depth control DDIM inversion [77] to the coarse rendering image. The prompt y_{inv}, which describes the coarse rendering, is used to obtain the inverted latent for each time step. During each denoising step, we denoise the inverted latent of the coarse rendering and the latent of the refined image, extracting their respective feature maps, $\boldsymbol{f}_t^c$ and $\boldsymbol{f}_t^f$, as well as their self-attention maps $\boldsymbol{A}_t^c$ and $\boldsymbol{A}_t^f$. This step is formulated as:

$$(\boldsymbol{f}_t^c, \boldsymbol{A}_t^c) \leftarrow \epsilon_\theta(\boldsymbol{x}_t^c, y_{inv}, D, t)$$
$$(\boldsymbol{f}_t^f, \boldsymbol{A}_t^f) \leftarrow \epsilon_\theta(\boldsymbol{x}_t^f, y, D, t)$$

Here, $\epsilon_\theta(\cdot)$ is the text-to-image diffusion model, specifically in our context, the Stable Diffusion XL [60] model. For the coarse rendering, the latent is denoted by $\boldsymbol{x}_t^c$, the inversion prompt by y_{inv}, and the depth map by D. For the refined image, the latent is represented by $\boldsymbol{x}_t^f$, and the prompt by y. Following Plug-and-Play [79], we replace the feature and self-attention maps of the enhanced image with those from the coarse input:

$$\boldsymbol{\epsilon}_{t-1}^f = \epsilon_\theta(\boldsymbol{x}_t^f, y, D, t; \boldsymbol{f}_t^f, \boldsymbol{A}_t^f)$$

Here $\epsilon_\theta(\cdot; \boldsymbol{f}_t^f, \boldsymbol{A}_t^f)$ represents the model with replaced feature and self-attention maps, and $\boldsymbol{\epsilon}_{t-1}^f$ is the prediction for the refined image. Replacement stops once the current time step is below the thresholds τ_f and τ_A. We set $\tau_A = 0.5$ and $\tau_f = 0.2$. The threshold is important because the feature/self-attention maps may contain undesired artifacts from coarse 3D reconstruction and mesh deformation.

B. Implementation Details

In this section, we provide implementation details of all our 6 tasks.

Pose Editing Pose editing is carried out by manually creating a skeleton for each 3D model and computing its skinning weights [6]. The object's pose is edited by adjusting the skeleton's bones. A text prompt is not required to describe the pose.

Rotation Rotation is achieved by spinning the 3D model around its centroid. This allows us to rotate the model at any angle and then render it back into a 2D image. However, it becomes challenging to discern the viewpoint (*e.g.*, front, back, or side), given only the coarse rendering image. Optional text prompts are helpful in guiding the denoising step and preventing the Janus Problem. If the rotation angle ranges from $[-45°, 45°]$, we add "*front view*" to the text prompt. For angle between $[135°, 225°]$, we append "*back view*" to the text prompt. For all other angles, we use "*side view*".

Translation Translation can be done by moving the 3D model within the 3D space in any direction and over any specific distance. As the translated model is rendered, the camera perspective adjusts accordingly. As illustrated in Fig. 6, moving the dog or the truck closer to the camera results in an enlarged image of the object, consistent with the camera's perspective.

Composition Our method allows for the addition of artist-created 3D objects to the scene. In Fig. 6, we insert various models into the scene. Despite the 3D models not being of high quality, our coarse-to-fine strategy significantly enhances their detail, as evident in the tiger example where the texture displays hair details and a realistic face in the final output, blending well with the environment. Note that these models are not used for fine-tuning during DreamBooth training. In certain cases, text prompts prove helpful in guiding the denoising step and supplementing our geometric guidance.

Carving Beyond mesh deformation, our method enables cutting and removing parts of the mesh through the use of molds. In Fig. 6, a moon-and-star-shaped mold is positioned against a pumpkin's surface. By calculating and excising the overlapping areas, the resulting mesh resembles a finely carved pumpkin in specific shapes.

Serial Addition Similar to composing elements, we can

Figure 13. Failure cases due to inaccurate reconstruction of texture, geometry, and color.

take meshes reconstructed from images and integrate them into the scene one by one. In Fig. 6, we adjust each fish and duck's size, pose and their orientation before adding it to the scene. Our approach realistically merges the coarse 3D fish models into the scene, maintaining a realistic appearance even with reflections on the water's surface.

C. *SculptingBench*

Our *SculptingBench* dataset comprises 28 edits applied to 15 images, encompassing each of the 6 editing tasks we have developed. The full dataset is illustrated in Fig. 14. These instances present significant challenges to current object editing techniques, thereby serving as an ideal platform for testing and developing precise object editing methods.

D. Limitation

Our method is an initial step towards integrating traditional geometric processing with advanced diffusion-based generative models for precise object editing. Yet, it has limitations. A significant challenge is the dependency on the quality of single-view 3D reconstruction, which is anticipated to improve over time. Additionally, mesh deformation often requires some manual efforts for model rigging. Future research might explore data-driven techniques [40] to automate this process. The output resolution of our pipeline also falls short of industrial rendering systems, and incorporating super-resolution methods could be a solution for future improvements. Another issue is the lack of background lighting adjustment, which undermines the realism of the scene; future work could benefit from integrating dynamic (re-)lighting techniques.

As demonstrated in Fig. 13, the reconstruction occasionally fails to produce detailed textures, leading to a blurred face in the top row example. Challenges also arise in mesh reconstruction and extraction. The middle row displays artifacts beneath the man's armpit, stemming from imprecise reconstruction in that region. In the bottom row example, wrong color reconstruction resulted in an less realistic final color in the output.

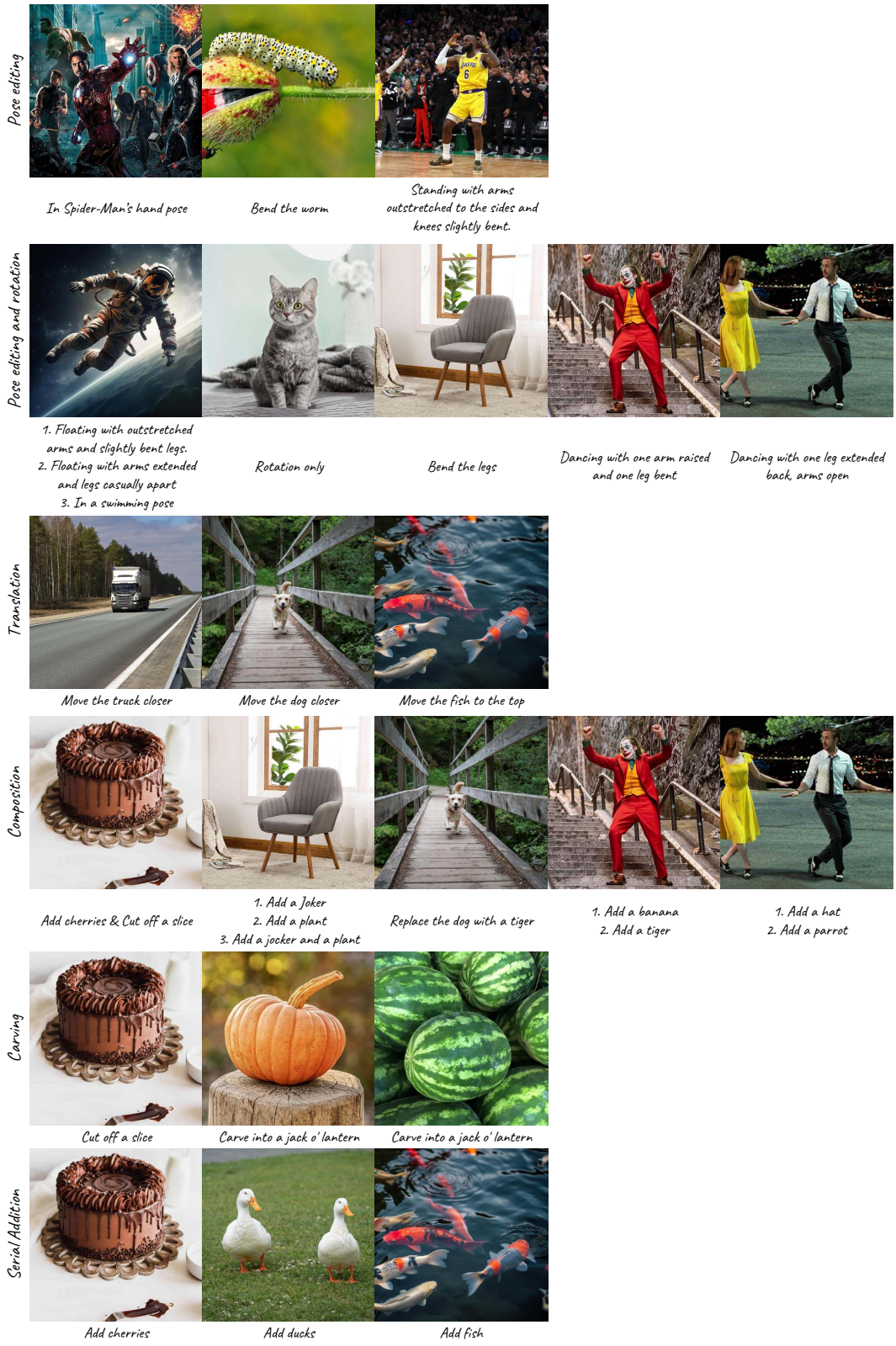

Figure 14. All 28 edits and 15 input images of our *SculptingBench*. We provide textual descriptions of the edits here. However, in practice we aim to make precise, quantifiable edits directly to 3D models, without relying on text prompts.