Mani-GS: Gaussian Splatting Manipulation with Triangular Mesh

Supplementary Material

Overview. The supplementary material has the following contents:

- More visual results
- Efficiency Analysis
- Implementation Details

A. More Visual Results

Demo Video. In order to further demonstrate the effectiveness of our methods, we have provided additional visual videos showcasing large deformation, soft body simulation, and local manipulation. These videos can be accessed through *project page*.

Soft Body Simulation. In addition to the visual results presented in the main paper, we also provide the geometry after simulation and rendering at different viewpoints in Figure 9. To improve the speed of the mesh simulation, we decimated the original mesh from 300K to 35K triangles. While this may result in some decrease in rendering quality due to the reduced number of triangles as well as Gaussians, it was necessary to ensure reasonable simulation speed.

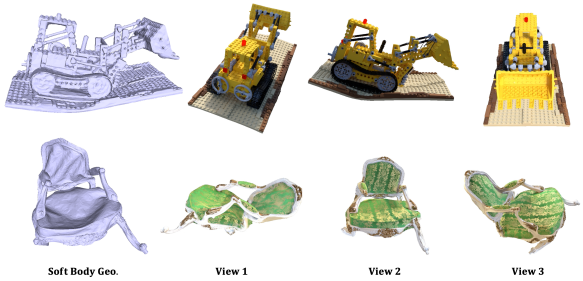

Figure 9. Visual results of softbody simulation at different viewpoints. The left column displays the geometry after simulation, while the right three columns showcase the rendering results from three different viewpoints.

More real case. We have also presented more manipulation results of real scenes in Figure 10. The top three images are from the DTU dataset, while the bottom two are from the Tanks and Templates dataset. Images within the dotted black rectangles are before manipulation, whereas those marked with red arrows or rectangles are after manipulation. The manipulation results presented in Figure 10 demonstrate that our approach can successfully transfer mesh manipulation to Gaussian-Splatting, resulting in accurate and visually appealing results.

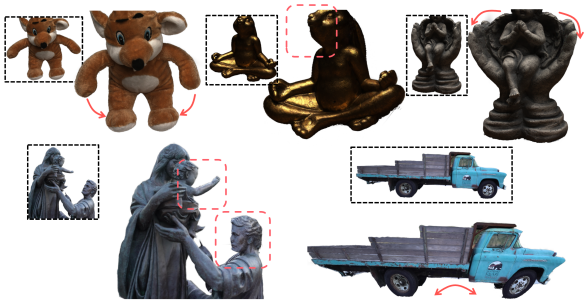

Figure 10. Manipulation rendering results on real object dataset. To highlight the deformed area, we have enclosed it within a red rectangle.

Table 4. Efficiency Analysis

	N=4	N=3	N=1	N=1	N=1
Triangles(K)	270	270	270	150	70
Points(K)	1080	810	270	150	70
Training (min)	16	13	**7**	5.5	4.5
Speed (FPS)	244	300	**452**	571	572
PSNR	36.36	**36.39**	36.27	35.86	34.52

B. Efficiency Analysis

The efficiency of 3DGS Binding training and rendering speed depends on the number of Gaussians, which is the product of the triangle number T and the Gaussians number for each triangle N. In Table 4, We first fixed T and tested different values of N. Our results indicate that N=3 leads to the best rendering quality while keeping a competitive rendering speed. When $N = 1$, the PSNR slightly decreased with a faster training and rendering speed.

We also evaluated the impact of underlying mesh resolution by testing meshes with different triangles (*270K, 150K, 70K*). As shown in Table 4, the rendering quality decreases while efficiency improves with decreasing mesh resolution.

Regarding the editing time, it primarily depends on the time cost of mesh editing. We use Blender for mesh editing, and in our experience, *local manipulation* and *large deformation* can be achieved instantly. *Soft body simulation* can be a more time-consuming process, as it depends on the simulation algorithm employed in Blender.

C. Implementation Details

C.1 Training Details of Mesh Extraction Stage

As outlined in our main paper, the first stage of our approach involves mesh extraction. While we utilize the NeuS[36] mesh as the foundation for binding Gaussians, we also explore extracting mesh from Gaussian-Splatting.

In this work, we try to extract triangular mesh using the Screened Poisson surface reconstruction [18] method from a trained Gaussian-Splatting model. We incorporate a normal attribute $\boldsymbol{n}$ for each 3D Gaussian and optimize the normal attribute with the pseudo-normal constraint.

The normal consistency is quantified as follows:

$$\mathcal{L}_n = \|\mathcal{N} - \tilde{\mathcal{N}}\|_2. \tag{8}$$

where $\mathcal{N}$ is the rendered-normal map, $\tilde{\mathcal{N}}$ is the pseudo-normal map computed from rendered depth map.

Besides the normal constraint $\mathcal{L}_n$, the ordinary L1 Loss and Structural Similarity Index (SSIM) loss are also incorporated into optimization by comparing the rendered image $\mathcal{C}$ with the observed image $\mathcal{C}_{gt}$. To address the issue of unwarranted 3D Gaussians in the background region, we employ a mask cross-entropy loss. This loss is defined as follows:

$$\mathcal{L}_{mask} = -B^m \log B - (1 - B^m) \log (1 - B), \tag{9}$$

where B^m denotes the object mask and B denotes the accumulated transmittance $B = \sum_{i \in N} T_i \alpha_i$.

Then all the loss terms can be summarized as follows:

$$\mathcal{L}_{stage1} = \lambda_1 \mathcal{L}_1 + \lambda_2 \mathcal{L}_{SSIM} + \lambda_3 \mathcal{L}_n + \lambda_4 \mathcal{L}_{mask}, \tag{10}$$

where $\lambda_1 = 1, \lambda_2 = 0.2, \lambda_3 = 0.01, \lambda_4 = 0.1$. We train this stage for 30K steps with adaptive density control, which is executed at every 500 iterations within the specified range from iteration 500 to 10K. Once the training stage is complete, we proceed with Screened Poisson surface reconstruction using the positions and normals of the Gaussians as input. The mesh extraction process takes less than 1 minute to complete.

In addition to mesh extraction, we also utilize Gaussian-Splatting Marching-Cube to extract the triangular mesh. Our approach involves sampling a grid with a resolution of $256 \times 256 \times 256$. For each sampling point, we identify its nearest Gaussian points. Sampling points that have the nearest Gaussians within a pre-defined distance threshold τ are assigned a density value of 1, while those that do not meet the threshold are assigned a density value of 0. τ is set to 0.01 in practice. The density threshold for Marching-Cube is set to 1e-4.

Based on the visual comparison, the overall mesh quality can be ranked as follows: NeuS $>$ Poisson Reconstruction $>$ Marching-Cube.

C.2 Training Details of Gaussian-Binding Stage

To ensure an accurate representation of each triangle, we bind N Gaussians to it. Prior to training, we initialize the positions of the Gaussians on the attached triangle. The N initialized position is calculated using a barycentric coordinate, with a predefined barycentric coordinate set of $[1/2, 1/4, 1/4]$, $[1/4, 1/2, 1/4]$, $[1/4, 1/4, 1/2]$. For the hyper-parameter β mentioned in main paper equation **(8)**, we set $\beta = 10$ in most cases, $\beta = 100$ in *Materials*.

In the Gaussian-Binding stage, we don't perform adaptive control because we find it doesn't influence the final performance. We also train 300K iterations in this stage with L1 loss, SSIM loss, and mask entropy loss. The overall loss in this can be summarized as follows:

$$\mathcal{L}_{stage2} = \lambda_1 \mathcal{L}_1 + \lambda_2 \mathcal{L}_{SSIM} + \lambda_3 \mathcal{L}_{mask}, \tag{11}$$

where $\lambda_1 = 1, \lambda_2 = 0.2, \lambda_3 = 0.1$.