

# MaGS: Reconstructing and Simulating Dynamic 3D Objects with Mesh-adsorbed Gaussian Splatting

## Supplementary Material

### Supplementary Material Overview

This supplementary material provides additional visuals and analyses to complement the main content of the paper. The sections are organized as follows:

- Section A includes implementation details for MPE-Net, RMD-Net, and RGD-Net.
- Section B describes hyperparameters. Specific details on opacity resets, densification, and MaGS parameters are included.
- Section C outlines the steps of the MaGS pipeline using pseudo-code (Algorithm 1). It covers mesh refinement and Mesh-adsorbed Gaussian optimization from input to output.
- Section D includes qualitative results comparing our method with existing approaches across datasets such as DG-Mesh, D-NeRF, and PeopleSnapshot. Visualizations in Figures A, B, and C highlight reconstruction accuracy and rendering quality.
- Section E showcases the results of simulations. Figures D, E, and F demonstrate complex object shapes and dynamic motions in a simulated environment.
- Section F presents a detailed quantitative analysis of MaGS. Metrics such as L1 loss, PSNR, and SSIM are reported in Tables A, B, C.
- Section G summarizes performance evaluations of MaGS in terms of FPS across scenes in the D-NeRF dataset. Results in Table D.
- Section H discusses limitations and future work.

### A. Implementation Details

In MPE-Net, we predict the mesh pose ( $\mathcal{E}_M$ ) and vertex-specific deformations ( $\mathcal{E}_V$ ) based on a coarse guide mesh. The input includes handle vertices ( $\mathcal{H}$ ), their normals ( $\mathcal{N}$ ), and the full mesh. Each vertex is encoded using positional encoding, generating a vertex embedding ( $\mathcal{E}_V$ ), which passes through fully connected layers with ReLU activations. These layers also produce a global mesh pose embedding ( $\mathcal{E}_M$ ). The model outputs both embeddings, allowing accurate deformation prediction without using temporal information.

We train RMD-Net using an MLP network  $F_\theta : (\mathcal{E}_M, \mathcal{E}_V) \rightarrow (\Delta v, \Delta q, \Delta s, \Delta \sigma, \Delta c)$ . A MLP  $F_\theta$  processes the input through  $D$  fully connected layers with ReLU activations, producing an initial feature vector. In the fourth layer, we concatenate this feature vector with the input. The resulting combined representation is then passed through

five additional fully connected layers, independently generating the outputs for  $\Delta v$ ,  $\Delta q$ ,  $\Delta s$ ,  $\Delta \sigma$ , and  $\Delta c$ .

The structure of RGD-Net closely resembles that of RMD-Net, utilizing a similar MLP architecture. The key difference is that RGD-Net also takes  $w$  as input. Additionally, instead of generating multiple outputs like RMD-Net, RGD-Net predicts a single output,  $\Delta w$ .  $\Delta w$  will be divided into two parts,  $\Delta b$  and  $\Delta h$ .

### B. Training Configuration

**Model Training Overview** The number of faces of the initialized mesh is approximately 1000. The number of iterations for optimizing is 40000.

**Optimization Hyperparameters** The optimization process leverages a combination of learning rates for the Mesh-adsorbed Gaussians components. Specifically, the  $b, h$ -parameters are optimized with a learning rate of  $lr = 0.00016$ , feature optimization uses  $lr = 0.0025$ , opacity is set to  $lr = 0.05$ , scaling employs  $lr = 0.005$ , and rotation uses  $lr = 0.001$ .

**Learning Rate Scheduling** The optimization of the  $b, h$ -parameters follows a learning rate scheduler. It begins with an initial rate of 0.00016 and gradually decays to 0.0000016 over the course of 40,000 iterations. This decay mechanism helps the model stabilize as training progresses.

**Densification Process** Densification occurs between iteration 100 and iteration 15,000. It is guided by a size threshold of 20 and a gradient threshold of 0.0002.

**Opacity Reset Mechanism** Opacity is periodically reset to maintain stability in the optimization process. This reset begins at iteration 300 and occurs at intervals of 3000 iterations.

**Loss Function Design** The loss function incorporates a structural similarity index (SSIM) term, weighted by  $\lambda_{ssim} = 0.2$ .

**Network Depth Configuration** For RMD-Net and RGD-Net, the network depth is controlled by the parameter  $D = 8$ . In contrast, MPE-Net uses a depth of  $E = 10$ .

**Background and Initialization** The background is set to black, with both the *random.bg* and *white.bg* options disabled (set to false). Furthermore, 50 Mesh-adsorbed Gaussians are randomly initialized on each facet, providing a starting point for the model’s spatial representation.

## C. Pseudo Code of Pipeline

The following pseudo-code outlines the key steps involved in the MaGS pipeline for mesh refinement and Gaussian optimization.

---

### Algorithm 1 MaGS Pipeline

---

**Require:** Video frames  $F$ , Initial mesh  $M_0$   
**Ensure:** Refined mesh  $M$  and Gaussians  $G$

```

1: procedure MAGS_PIPELINE( $F, M_0$ )
2:   // Step 1: Initialize Gaussians
3:    $G \leftarrow \text{INITIALIZEGAUSSIANS}(M_0)$ 
4:   for each  $f \in F$  do
5:     // Step 2: Extract feature embeddings
6:      $(E_M, E_V) \leftarrow \text{MPE\_NET}(M_0, f)$ 
7:     // Step 3: Predict mesh and Gaussian updates
8:      $(\Delta b, \Delta h, \Delta params) \leftarrow$ 
        $\text{RMD\_NET}(E_M, E_V)$ 
9:      $\Delta w \leftarrow \text{RGD\_NET}(E_M, E_V, G)$ 
10:    // Step 4: Update mesh and Gaussians
11:     $M \leftarrow M + \Delta v$ 
12:     $G \leftarrow G + \text{INTERP}(\Delta params, G, \Delta b, \Delta h, M)$ 
13:    // Step 5: Render and compute loss
14:     $I \leftarrow \text{RENDERGAUSSIANS}(G)$ 
15:     $L \leftarrow \text{LOSS}(I, f.\text{gt})$ 
16:    // Step 6: Backpropagate loss
17:     $\text{BACKPROPAGATE}(L, \{\text{MPE\_NET}\})$ 
18:     $\text{BACKPROPAGATE}(L, \{\text{RMD\_NET}\})$ 
19:     $\text{BACKPROPAGATE}(L, \{\text{RGD\_NET}\})$ 
20:  end for
21:  // Step 7: Final refinement
22:   $M, G \leftarrow \text{REFINE}(M, G)$ 
23:  return  $M, G$ 
24: end procedure
```

---

## D. Additional Visualizations of Reconstruction

In this section, we present additional visualizations and comparisons to validate our findings further and demonstrate the performance of our methods.

Figure A presents additional qualitative results showcasing the performance of our method on the DG-Mesh dataset. The comparison demonstrates that our approach achieves mesh reconstruction close to the ground truth, providing higher accuracy in the reconstructed meshes than previous methods with fewer facets.

Figure B presents an L1 loss visualization on the D-NeRF dataset, comparing the deformation predictions of our model with those of existing methods. This detailed qualitative comparison demonstrates that our approach achieves the most accurate rendering, highlighting its superior performance.

Similarly, Figure C illustrates an L1 loss visualization of the results on the PeopleSnapshot dataset, providing insights into rendering accuracy in real-world scenarios. The visualization confirms that our method is highly accurate, further validating its suitability for human pose tasks.

## E. Additional Visualizations of Simulation

Figure D illustrates the results of simulations conducted on the Lego scene, demonstrating how our method effectively handles complex object shapes and dynamic motion in a simulated environment. These visualizations highlight the robustness of our approach to managing intricate geometries and movements, further validating its effectiveness in challenging scenarios.

Similarly, we conducted simulations on the Horse scene with dragging-based editing. The horse’s legs undergo significant deformations, with corresponding movements observed in the body. Despite these deformations, the textures in the detailed regions are preserved throughout the simulation, as seen in Figure E.

Furthermore, we applied gravity and collision simulations to the Beagle from DG-Mesh and the Mutant from D-NeRF. After repositioning the objects, gravitational forces were applied. The results, visualized in Figure F, demonstrate the dynamics of falling, collision, and rebound behaviors. For clarity, we selected one frame every five frames for presentation, with the actual simulations exhibiting smoother and more coherent transitions.

## F. Detail Quantitative Comparisons

In this section, we present a comprehensive quantitative analysis of the performance of our proposed methods compared to prior approaches. Detailed experimental results from ablation studies and scene-specific evaluations are included to validate the robustness and effectiveness of our approach.

Table A summarizes the results of the ablation experiments conducted on the D-NeRF dataset, reporting metrics such as L1 loss, PSNR, and SSIM for each scene. These results demonstrate that the key components of our method (RMD-Net, RGD-Net, and Hovering) significantly improve the model’s performance.

Table B provides an analysis of the DGMesh dataset. The results indicate that our method consistently achieves higher PSNR and EMD scores across all scenes compared to existing methods, showcasing its adaptability and precision in handling complex mesh structures and dynamic motions.

It is worth noting that the *Torus2Sphere* scene involves objects undergoing significant topological changes. The provided ground-truth meshes for this scene do not guarantee consistent vertex and face correspondences across

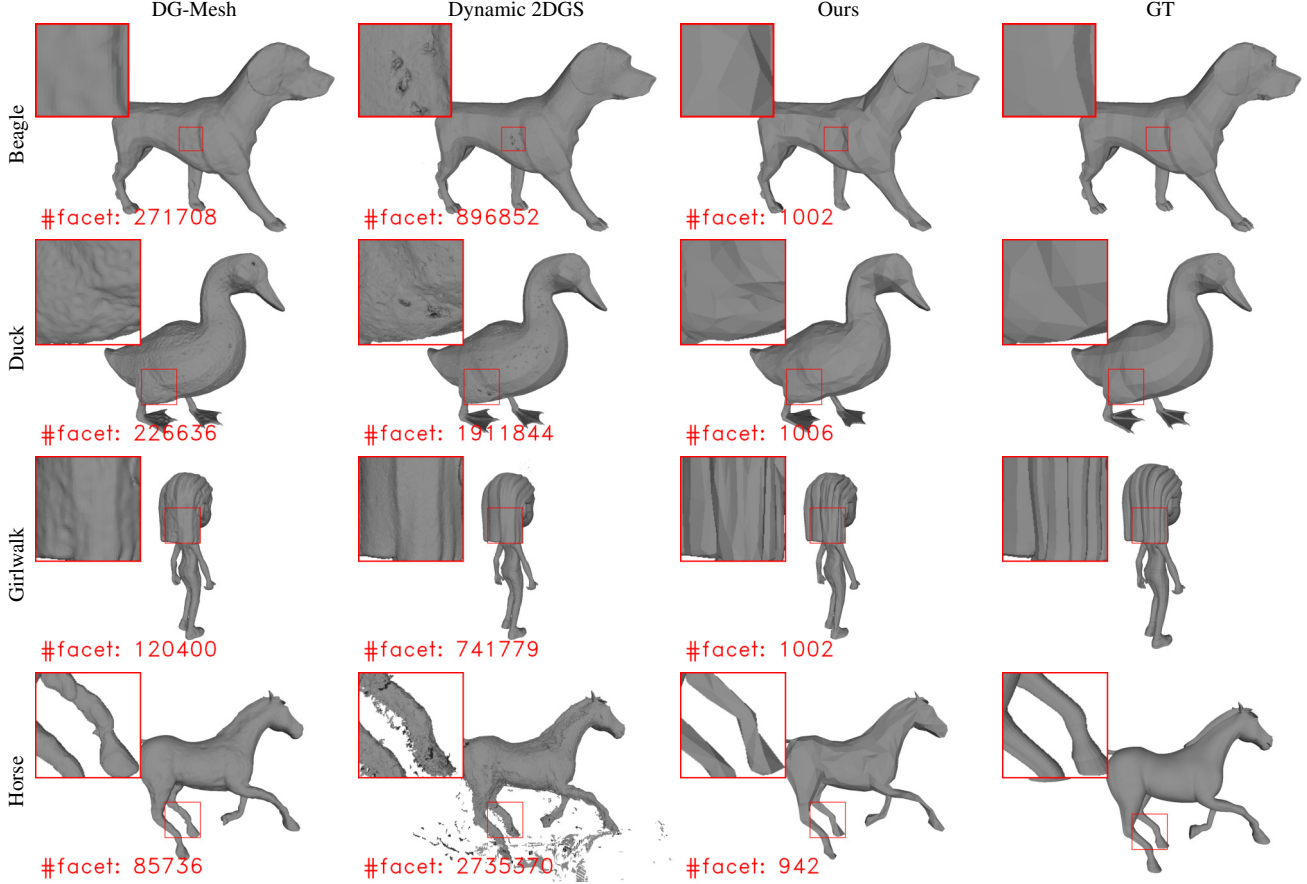


Figure A. **Qualitative Comparison on the DG-Mesh Dataset.** MaGS achieves superior rendering and simulation results with better mesh quality, while using several orders of magnitude fewer facets, demonstrating its efficiency and effectiveness.

frames. This makes it unsuitable for evaluating methods that preserve inter-frame vertex and face correspondences. While our approach achieved higher EMD and PSNR scores for this scene, we have excluded it from the main quantitative tables due to these inconsistencies.

As mentioned in the main text, the test set for the Lego scene contains temporal and image inconsistencies. Yang *et al.* provided a corrected version of the Lego, and we conducted tests on this corrected dataset. The results shown in Table C demonstrate that our method still outperforms other approaches on the Lego scene.

## G. Performance Benchmark

Table D presents the performance evaluation of frames per second (FPS) with respect to the number of 3D Gaussians across various scenes in the D-NeRF dataset, evaluated at a resolution of  $400 \times 400$ . The experiments were conducted using a single RTX 4090 GPU over 40,000 training iterations, with FPS measured on the full set of training frames. The results highlight significant variability in FPS

across different scenes, influenced by the number of Gaussians (denoted as “Number of Gaussians (k)”). On average, MaGS achieves an FPS of 129.19 with 84.8k Gaussians across all evaluated scenes, demonstrating its efficiency in handling diverse 3D Gaussian configurations.

## H. Limitations and Future Work.

Despite these successes, MaGS has limitations: 1) it struggles with multi-object interaction scenarios; 2) it requires video data with diverse viewpoints for reliable mesh construction—an inherent challenge for mesh-based methods. Future work will focus on addressing these challenges by integrating generative models to enhance MaGS’s capabilities and robustness under uncertain or incomplete viewpoint conditions.

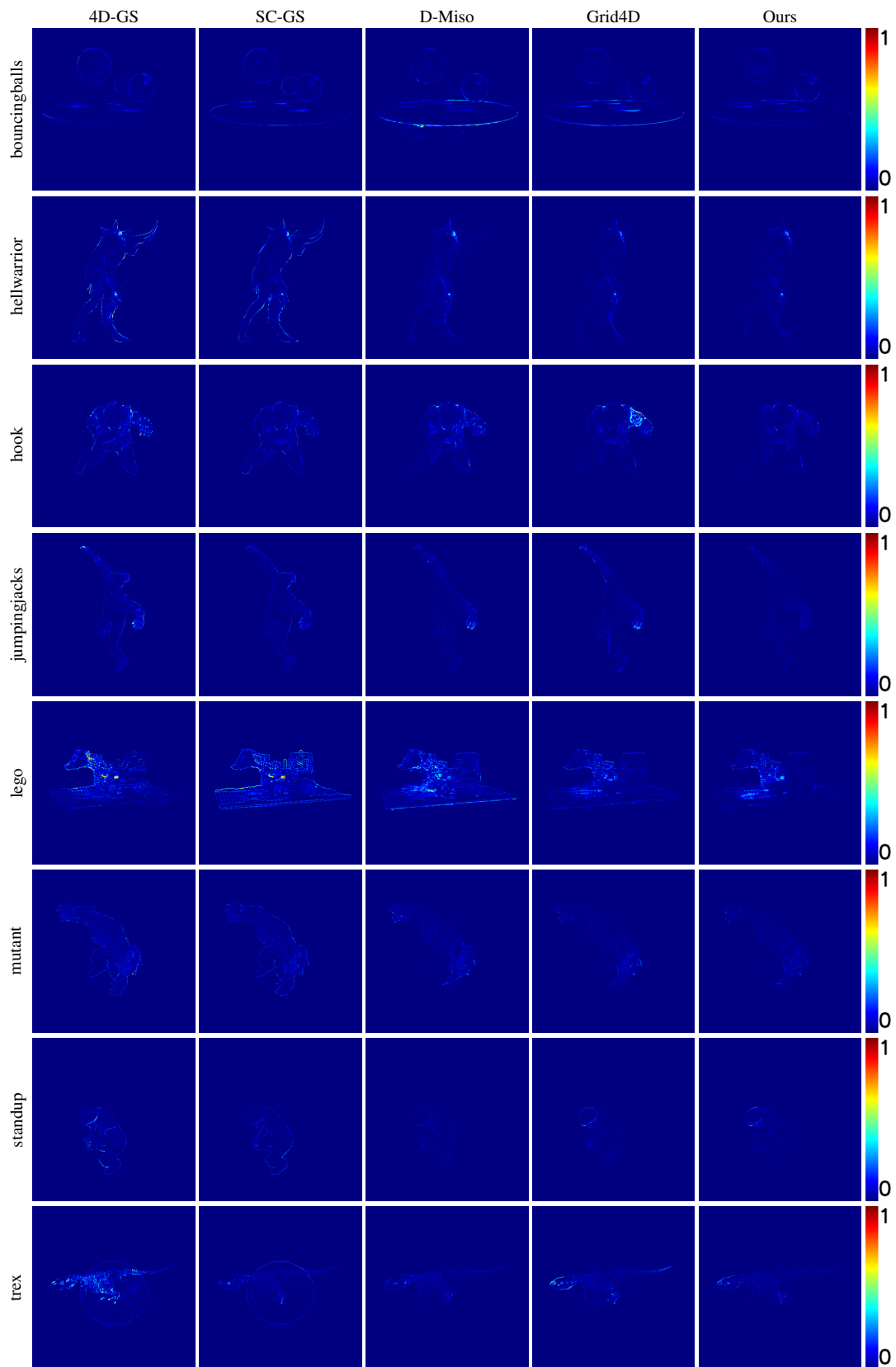


Figure B. **Visualization of L1 Loss (between rendered and GT image) Qualitative Results on D-NeRF.** We compare MaGS with 4D-GS, SC-GS, D-Miso, and Grid4D. All images are processed using the same pseudo-color conversion algorithm (CV2's COLORMAP-JET).

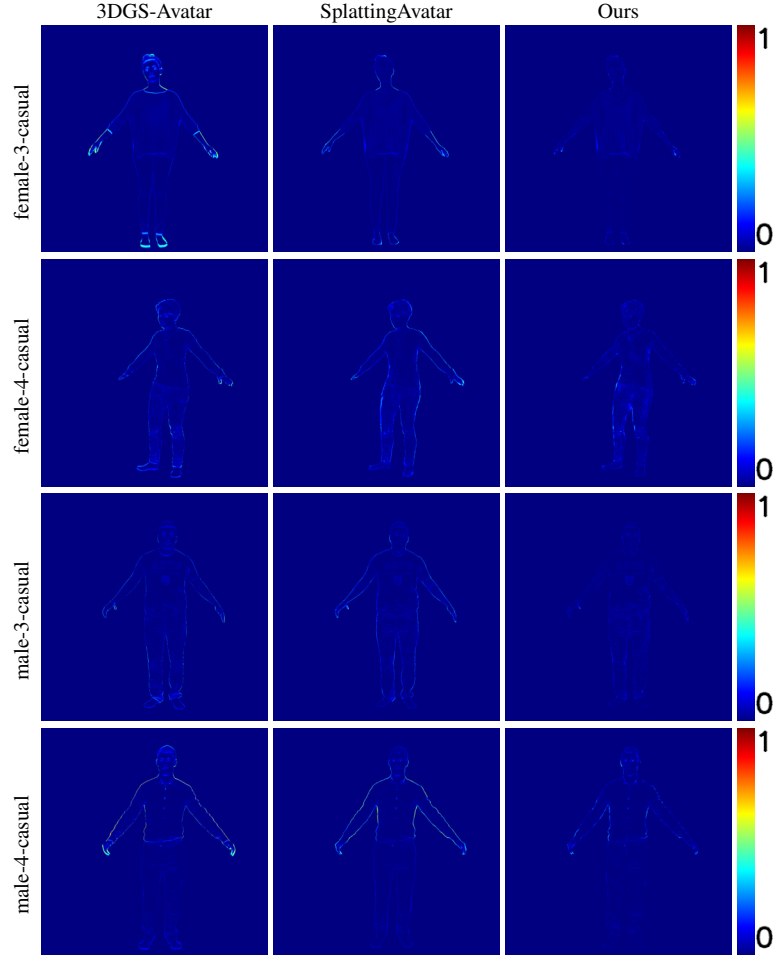


Figure C. **Visualization of L1 Loss (between rendered and GT image) Qualitative Results on PeopleSnapshot.** We compare MaGS with 3DGS-Avatar and SplattingAvatar. All images are processed using the same pseudo-color conversion algorithm (CV2’s COLORMAP-JET).

Table A. **Ablation experiments on the D-NeRF dataset.**

Method	Bouncingballs			Hellwarrior			Hook			Jumpingjacks		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
MaGS w/o MDF and RDF	40.31	0.9973	0.0065	42.38	0.9939	0.0137	37.67	0.9963	0.0098	40.30	0.9984	0.0051
MaGS w/o Hover	39.29	0.9966	0.0076	42.05	0.9937	0.0147	38.80	0.9972	0.0074	41.61	0.9988	0.0036
MaGS w/o RDF	40.80	0.9975	0.0058	42.86	0.9947	0.0125	40.28	0.9981	0.0055	42.30	0.9990	0.0031
MaGS	41.97	0.9976	0.0055	43.69	0.9957	0.0098	41.23	0.9984	0.0049	44.29	0.9993	0.0022
Method	Mutant			Standup			Trex			Average		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
MaGS w/o MDF and RDF	44.31	0.9990	0.0025	44.97	0.9991	0.0026	38.07	0.9976	0.0047	41.14	0.9974	0.0064
MaGS w/o Hover	43.49	0.9989	0.0029	47.98	0.9996	0.0014	39.85	0.9989	0.0036	41.87	0.9977	0.0059
MaGS w/o RDF	44.70	0.9991	0.0023	48.66	0.9997	0.0011	41.28	0.9992	0.0027	42.98	0.9982	0.0047
MaGS	46.42	0.9996	0.0019	49.16	0.9997	0.0010	41.65	0.9993	0.0025	44.06	0.9985	0.0040



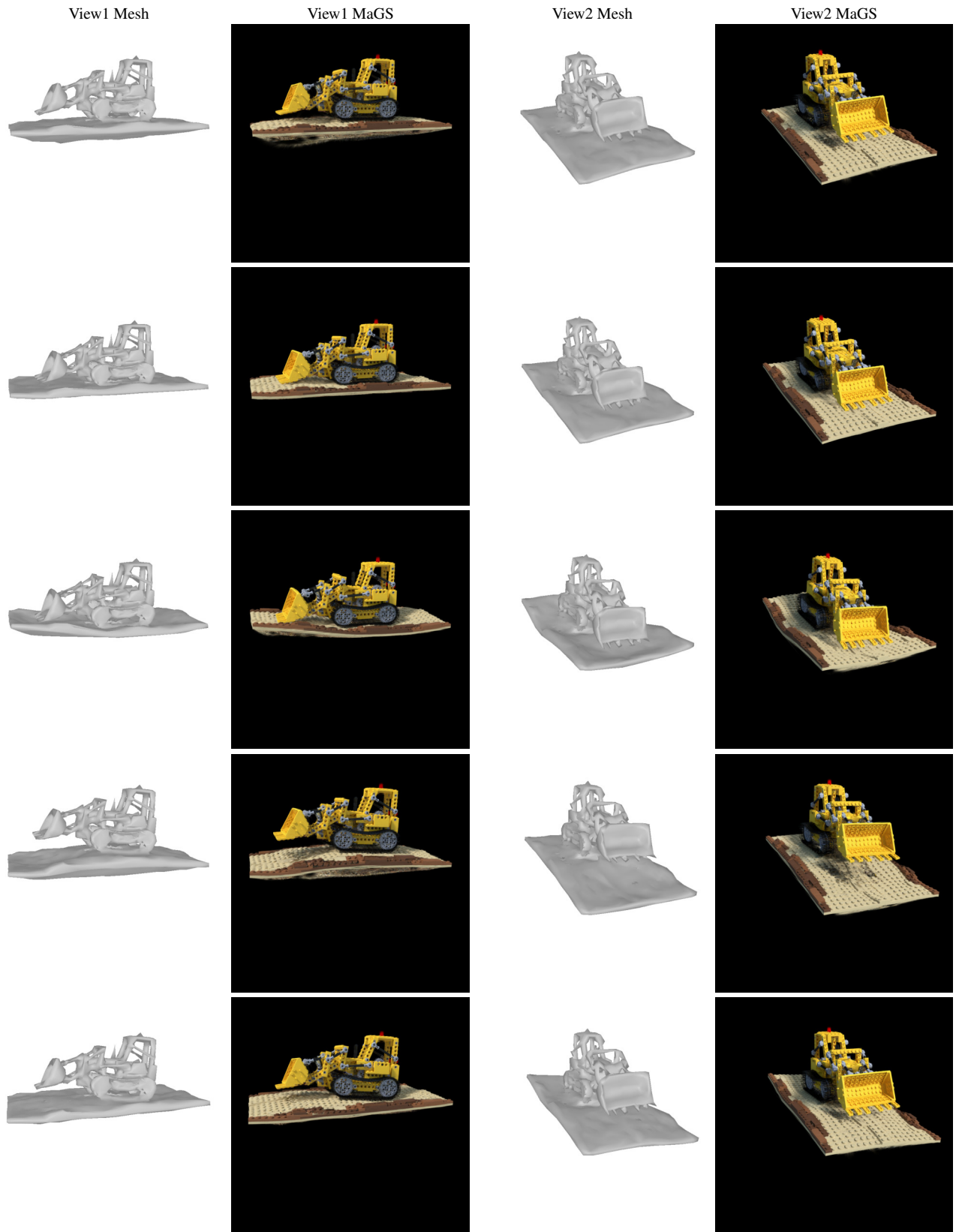


Figure D. **Soft-body Drop Simulation on Lego.** MaGS effectively simulates the texture and deformation of objects during the collision process when they fall to the ground.

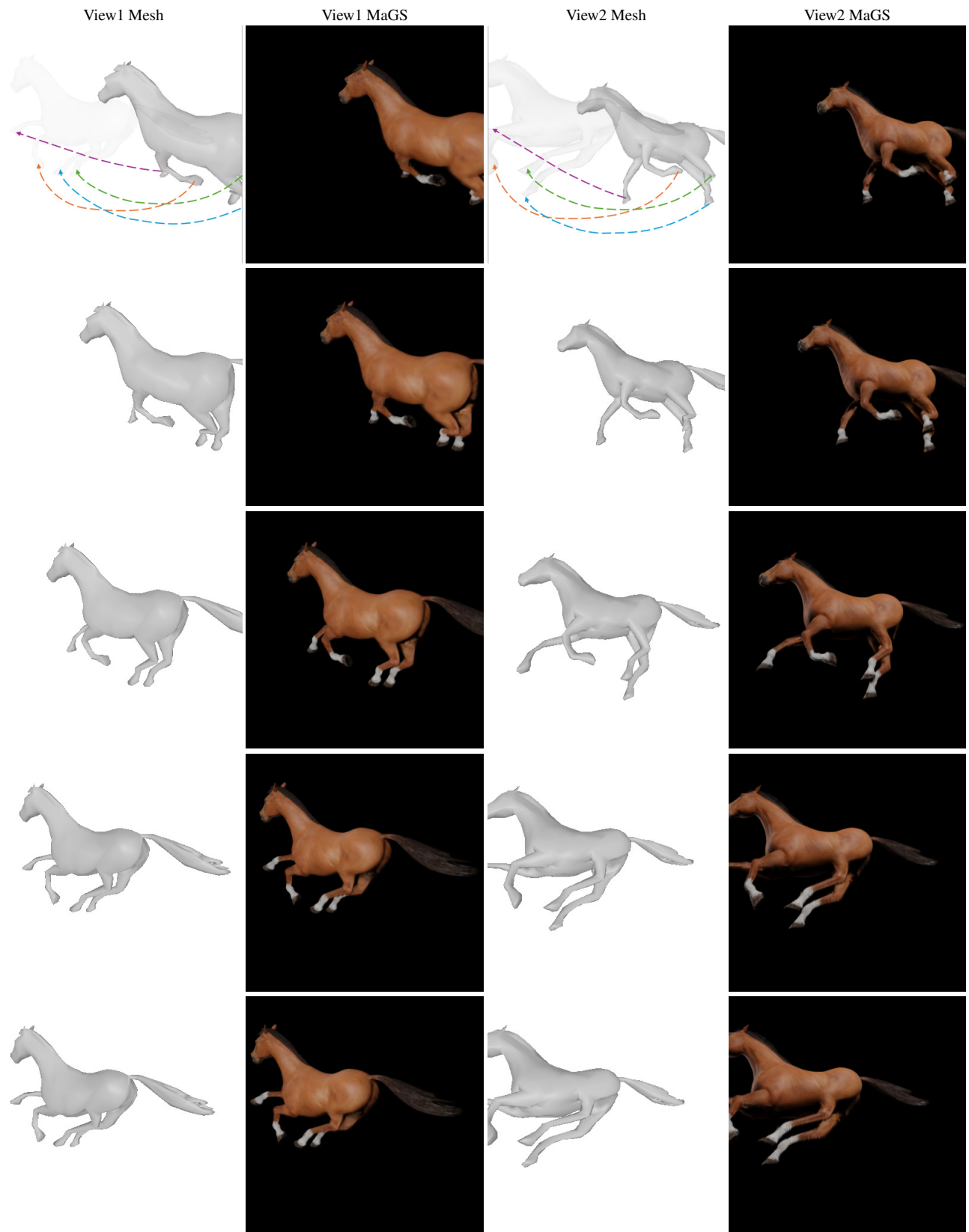


Figure E. **Drag Editing Simulation on Horse.** The arrows indicate the vectors of the user's dragging forces. MaGS effectively preserves the geometric priors of objects during deformation.

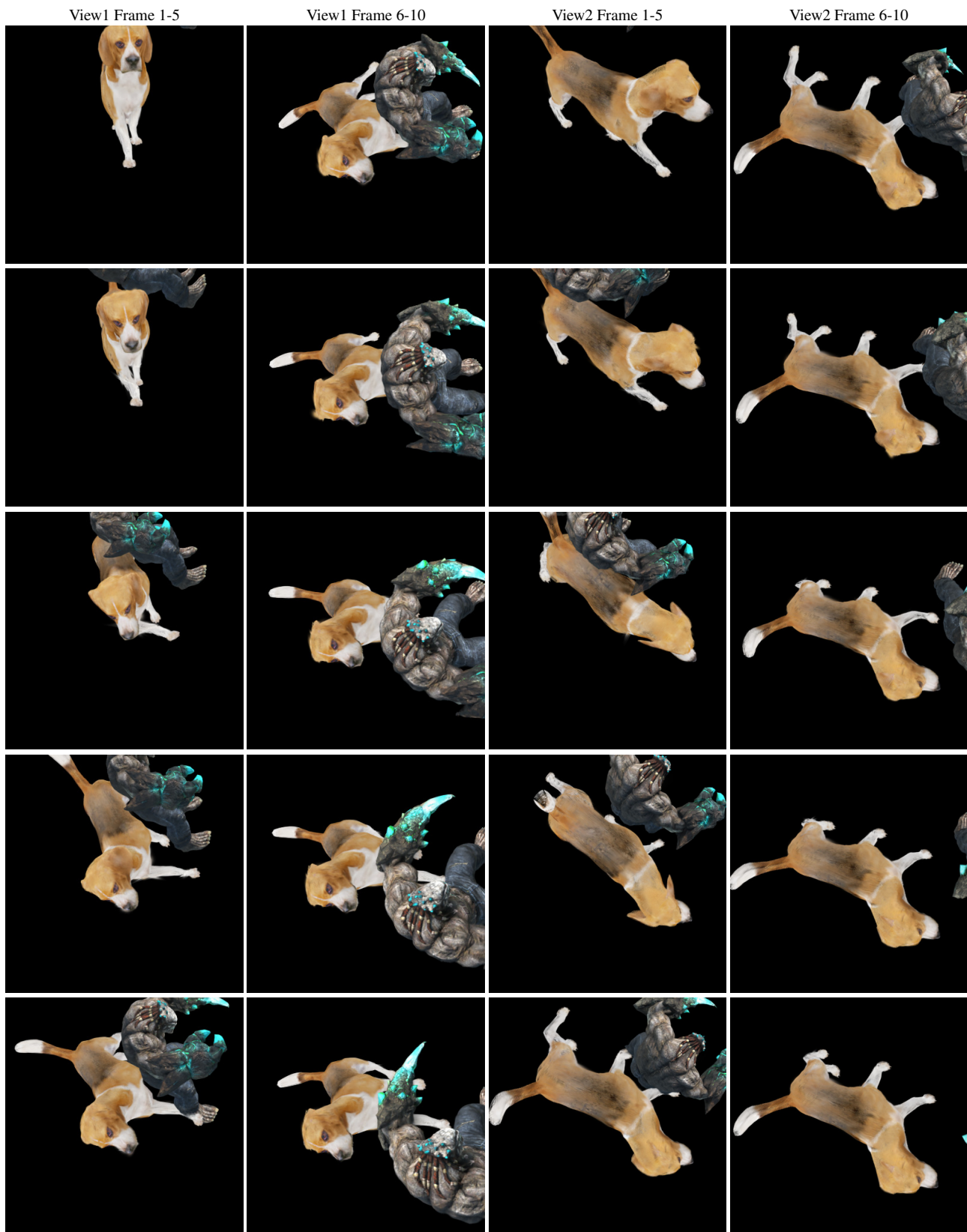


Figure F. **Gravity and Collision Simulation of Beagle and Mutant.** MaGS effectively simulates the physical-plausible deformations during the interaction of multiple objects.



Table B. Quantitative Results on DG-Mesh.

Methods	Beagle			Girlwalk			Duck		
	CD↓	EMD↓	PSNR↑	CD↓	EMD↓	PSNR↑	CD↓	EMD↓	PSNR↑
D-NeRF	1.0010	0.1490	34.47	0.6010	0.1900	28.63	0.9340	0.0730	29.79
K-Plane	0.8100	0.1220	38.33	0.4950	0.1730	32.12	1.0850	0.0550	33.36
HexPlane	0.8700	0.1150	38.03	0.5970	0.1550	31.77	2.1610	0.0900	32.11
TiNeuVox-B	0.8740	0.1290	38.97	0.5680	0.1840	32.81	0.9690	0.0590	34.33
DG-Mesh	0.6390	0.1170	33.41	0.7260	0.1360	32.91	0.7900	0.0470	32.26
DynaSurfGS	0.6090	0.1100	40.74	0.4430	0.1280	33.31	0.8060	0.0470	36.31
Dynamic 2D Gaussians	0.5440	0.1220	41.94	0.3240	0.1290	41.17	1.0400	0.0920	38.95
Ours	0.8252	0.1115	43.11	0.7216	0.1307	44.78	0.8070	0.0681	42.03

Methods	Horse			Bird			Torus2sphere		
	CD↓	EMD↓	PSNR↑	CD↓	EMD↓	PSNR↑	CD↓	EMD↓	PSNR↑
D-NeRF	1.6850	0.2800	25.47	1.5320	0.1630	23.85	1.7600	0.2500	24.23
K-Plane	1.4800	0.2390	28.11	0.7420	0.1310	23.72	1.7930	0.1610	31.21
HexPlane	1.7500	0.1990	26.80	4.1580	0.1780	22.19	2.1900	0.1900	29.71
TiNeuVox-B	1.9180	0.2460	28.16	8.2640	0.2150	25.55	2.1150	0.2030	28.76
DG-Mesh	0.2990	0.1680	30.64	0.5570	0.1280	27.91	1.6070	0.1720	11.84
DynaSurfGS	0.2960	0.1450	28.68	1.6310	0.1380	26.88	1.6750	0.1710	29.13
Dynamic 2D Gaussians	0.3910	0.1770	31.92	0.3280	0.1100	28.03	2.4790	0.1640	30.17
Ours	0.2510	0.1525	39.09	0.7263	0.0900	34.79	3.3167	0.1227	34.34

Table C. Quantitative Results on Lego

Methods	PSNR↑	MS-SSIM↑	VGG-LPIPS↓
4D-GS	28.72	0.9822	0.0368
D-MiSo	28.43	0.9810	0.0461
SP-GS	30.83	0.9864	0.0221
Deformable-GS	33.07	0.9794	0.0183
SC-GS	33.11	0.9886	0.0178
Grid4D	33.24	0.9938	0.0132
Ours	34.56	0.9945	0.0114

Table D. **Frames Per Second (FPS) Evaluation with Respect to the Number of 3D Gaussians**

<b>D-NeRF Dataset (<math>400 \times 400</math>), Single RTX 4090, 40,000 Training Iterations, FPS Evaluated on Full Training Frames</b>		
<b>Scene</b>	<b>FPS</b>	<b>Number of Gaussians (k)</b>
Jumping Jacks	122.68	88.6
Bouncing Balls	132.62	73.5
Hell Warrior	139.45	66.6
Hook	119.95	86.1
Stand-up	195.59	42.9
T-Rex	134.58	81.2
Lego	103.09	98.5
Mutant	65.59	140.8
<b>Average</b>	<b>126.69</b>	<b>84.8</b>