

Vid2Sim: Generalizable, Video-based Reconstruction of Appearance, Geometry and Physics for Mesh-free Simulation

Supplementary Material

This supplementary material covers the following sections: More Implementation Details (Sec. 1); More Results on Dynamic Reconstruction (Sec. 2); Generalization Capability (Sec. 3). Please refer to our supplementary video for a more comprehensive overview,

1. More Implementation Details

1.1. Large Video Vision Transformer

The pipeline of our Large Video Vision Transformer is shown in Fig. 1. In our framework, we fine-tune the backbone network, VideoMAE [7], which is pre-trained on 16-frame videos at a resolution of 224×224 . To adapt it to a higher resolution (448×448 in our setting), we interpolate the pre-trained positional embeddings to align with the updated number of input tokens. The output tokens are averaged across all the patches before being sent into the regression MLPs. The regression MLPs for predicting E and ν are identical and with widths of [768, 512, 256, 128, 1]. The regression MLP for predicting $\hat{\theta}_{lbs}$ has widths of [768, 650, 650, 650, 650], where the width of the last layer is equal to the number of trainable parameters for a linear layer. We demonstrate in Tab. 1 that it is better to predict only the last layer of $\hat{\theta}_{lbs}$ and keep the first 7 layers fixed for consistency with the optimization stage (Stage II) than to predict full layers in our task. This is because Hypernetwork predicts $\sim 30k$ network parameters for full-layer LBS, making training much more difficult than our one-layer prediction design. We use GELU [4] as the activation function for all regression MLPs.

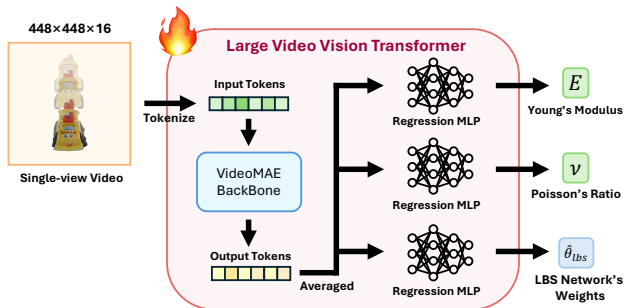


Figure 1. Detailed pipeline of the large video vision transformer.

1.2. LBS and Jacobian Network

The implementation of the LBS and Jacobian network is visualized in Fig. 2. Specifically, the LBS network com-

	PSNR \uparrow	SSIM \uparrow	FoVDP \uparrow
One-layer prediction (Ours)	28.83\pm3.06	0.954\pm0.014	7.907\pm0.859
Full-layer prediction	28.53 \pm 3.21	0.953 \pm 0.015	7.782 \pm 0.886
GT (data-free train)	29.40 \pm 2.59	0.957 \pm 0.012	8.169 \pm 0.579

Table 1. Quantitative results in Dynamic Reconstruction across different LBS prediction settings using the same optimized geometry and physical parameters for fairness.

	Forward Time	Backward Time	$\frac{1}{N} \sum_{i=1}^N \mathbf{J}_{\theta}^i - \mathbf{J}_{gt}^i _2^2$
4 blocks w/o PE	0.00081s	0.00158s	9.22×10^{-7}
2 blocks	0.00052s	0.00099s	5.12×10^{-7}
4 blocks (Ours)	0.00086s	0.00166s	4.04×10^{-8}
GT Jacobian	6.51398s	0.00014s	-

Table 2. Speed (time per iteration) and average accuracy across different Neural Jacobian models. All the values are tested under the setting of 2000 points & 10 handles on one NVIDIA-RTX-4090 GPU.

prises 8 linear layers with a constant layer width of 64 and ELU [3] activation function. We observe that the neural Jacobian predominantly focuses on learning to predict high-frequency features, rather than the low-frequency signals typically modeled by the LBS prediction network. This insight motivates us to adopt a design for predicting the Jacobian that differs from the standard MLP architecture used in the LBS network, where we incorporate positional encoding into the input to capture the high-frequency features effectively. The input positions are embedded into a 512-dimensional space using positional encoding. The model comprises four residual blocks, each containing two linear layers. The first two residual blocks have a layer width of 512, while the last two have a layer width of 1024. The output is projected with a linear layer from the features. We use the GELU [4] activation function in the Jacobian network. We found that 4 blocks with positional encoding are sufficient to predict the Jacobian that is accurate enough for simulation, so we didn't scale up it further to save data-free training time. We report the speed-accuracy trade-off for different Neural Jacobian models in Tab. 2. Note that the time cost of Neural Jacobian is only meaningful in its data-free training, which contains 10k iterations. It can be ignored in the joint optimization of Stage II.

The LBS and Jacobian networks are first trained in a data-free manner, supervised by randomly sampled \mathbf{X} and \mathbf{z} , inspired by [1, 6]. The LBS network is optimized by minimizing an elastic loss and orthogonal regularization loss.

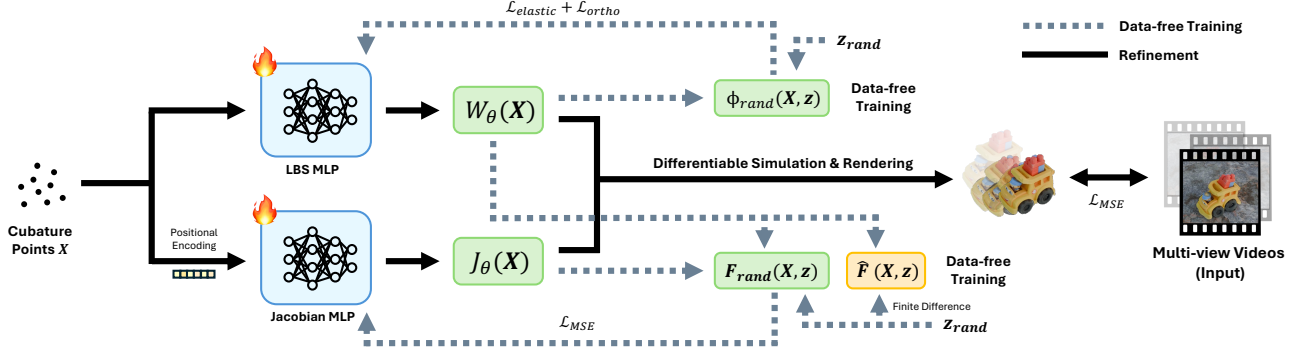


Figure 2. Network structure of LBS network and Jacobian network.

		backpack	bell	blocks	bus	cream	elephant	grandpa	leather	lion	mario	sofa	turtle	Mean
PSNR \uparrow	PAC-NeRF	18.03	21.74	21.67	19.05	19.81	20.68	20.20	19.48	20.67	17.06	19.60	22.09	20.01
	Spring-Gaus	17.35	21.04	22.93	19.77	24.80	20.97	21.76	19.28	21.32	20.51	21.55	22.42	21.14
	GIC	18.22	19.33	18.26	20.23	23.94	21.50	20.45	20.76	18.53	21.17	23.38	22.60	20.70
	Ours (full)	26.59	27.26	31.29	25.64	33.85	27.95	24.09	31.11	25.89	26.54	27.82	30.81	28.24
SSIM \uparrow	PAC-NeRF	0.882	0.956	0.935	0.900	0.900	0.924	0.941	0.929	0.931	0.932	0.918	0.933	0.924
	Spring-Gaus	0.866	0.945	0.936	0.899	0.918	0.921	0.950	0.924	0.931	0.922	0.913	0.931	0.921
	GIC	0.879	0.938	0.918	0.903	0.901	0.924	0.943	0.937	0.917	0.930	0.920	0.934	0.920
	Ours (full)	0.940	0.964	0.971	0.935	0.951	0.955	0.953	0.975	0.949	0.951	0.942	0.973	0.955
FoV/DP \uparrow	PAC-NeRF	5.873	6.514	6.803	6.330	4.695	6.681	6.516	6.437	6.581	3.989	5.779	6.943	6.095
	Spring-Gaus	5.535	6.945	6.920	6.344	5.970	6.482	7.045	6.260	6.766	6.201	6.296	7.011	6.481
	GIC	5.959	6.129	5.947	6.394	5.805	6.921	6.930	6.696	6.023	7.029	6.905	6.939	6.473
	Ours (full)	8.331	7.640	8.921	7.898	8.900	8.398	7.020	9.053	7.659	8.015	7.912	9.000	8.229

Table 3. Quantitative comparison with previous methods on dynamic reconstruction (novel views).

		backpack	bell	blocks	bus	cream	elephant	grandpa	leather	lion	mario	sofa	turtle	Mean
$\log(E)$	PAC-NeRF	3.28	1.08	4.02	3.30	3.22	3.05	2.99	1.20	2.34	3.37	0.20	1.94	2.50
	GIC	1.16	2.87	2.12	1.93	2.13	1.53	0.42	3.45	2.85	1.82	0.65	3.18	2.01
	Ours (full)	0.69	0.15	0.54	0.26	0.95	0.18	1.07	0.73	0.48	0.50	0.18	0.44	0.51
ν	PAC-NeRF	0.21	0.23	0.33	0.16	0.12	0.06	0.36	0.26	0.14	0.33	0.30	0.01	0.21
	GIC	0.11	0.24	0.29	0.18	0.08	0.24	0.26	0.02	0.14	0.22	0.01	0.08	0.16
	Ours (full)	0.10	0.10	0.07	0.06	0.11	0.05	0.02	0.06	0.05	0.06	0.08	0.02	0.06

Table 4. Mean Absolute Error (MAE) among baselines and our method on physical property predictions.

The Jacobian network is optimized by minimizing the L2 loss between the predicted deformation gradient $\mathbf{F}(\mathbf{X}, \mathbf{z})$ and the estimated $\hat{\mathbf{F}}(\mathbf{X}, \mathbf{z})$ from the finite difference.

The two networks are then jointly trained along with physical parameters according to the observed multi-view videos, where we only minimize the L2 loss between simulated animations and the observed multi-view videos, as described in Sec. 4.3 in the main paper.

1.3. Boundary Condition Implementation

We follow [6] to implement boundary conditions with incremental potential contact for handling collision, the constraints are formulated with barrier functions that provide

extra potential energy. For example, our floor barrier in the dynamic reconstruction and the future state prediction task uses $E_f = 10^5 \times \sum_{i=1}^N [\max(0, h_f - h_i)]^2$ as potential energy, where E_f is part of the external energy (See Eq. 2 in the main paper). Barrier functions can be very flexible in our method, and we provide more examples in Sec. 3.2.

2. More Results on Dynamic Reconstruction

In Sec. 2.1, we provide a comprehensive investigation by showcasing additional qualitative results of dynamic reconstruction and future state prediction across baselines, our Stage I model, and our full model. In Sec. 2.2, we evaluate

the performance of dynamic reconstruction on novel views. Additionally, we evaluate the prediction of physical properties E and ν in Sec. 2.3.

2.1. More Qualitative Results on Dynamic Reconstruction and Future States Prediction

As illustrated in Fig. 3 and Fig. 4, our model demonstrates remarkable physics-aware dynamic reconstruction quality compared to existing methods [2, 5, 8] that suffer from reconstructing blurry textures and incorrect dynamics due to the use of dynamic representations and symplectic solver. This is further evidenced by real-world test cases presented in Fig. 5, where the reconstruction results of the SOTA method Spring-Gaus [8] collapses when hitting the ground plane, while ours successfully capture the physical dynamics and produce higher realistic results.

2.2. Evaluation on Novel View Synthesis

We further evaluate the performance of our method on novel view synthesis by randomly sampling 6 novel views for each synthetic test case and evaluate the dynamic reconstruction performance among our method and baselines. We show qualitative results in Fig. 6 and quantitative results in Tab. 3, where our method consistently outperforms other models.

2.3. Evaluation on Physical Parameters Estimation

Next, we evaluate the Mean Absolute Error (MAE) on the estimated $\log(E)$ and ν in the Neo-Hookean elastic model used by PAC-NeRF [5], GIC [2] and our method. As shown in Tab. 4, our method outperforms all the other approaches in most cases while showing its competitive performance on the remaining samples, which validates the effectiveness of our model on physical property estimation.

3. Generalization Capability

We provide more simulation results on changed materials in Sec. 3.1 and provide additional simulation results on different boundary conditions in Sec. 3.2.

3.1. Generalized to Different Materials

Although our method mainly focuses on reconstructing elastic objects in this paper, our framework can be generalized to materials characterized by various constitutive models. Here, we show simulation results regarding three different materials: Elasticity, Plasticine, and Sand following [2, 5]. The qualitative results are shown in Fig. 7, where different materials are simulated precisely as our method is combined with different constitutive models effectively.

In order to compute the potential energy $E_{\text{potential}}$ for simulation, we derive the corresponding energy density function $\Psi(\mathbf{F})$ for each constitutive model below.

Elasticity. The energy density function can be formulated as

$$\Psi(\mathbf{F}) = \frac{\mu}{2}[\text{tr}(\mathbf{F}^\top \mathbf{F}) - d] - \mu \ln(J) + \frac{\lambda}{2} \ln^2(J) \quad (1)$$

where $d = 3$ is the space dimension, \mathbf{F} is the deformation gradient and J is the determinant of \mathbf{F} , μ and λ are Lamé parameters related to Young’s modulus E and Poisson’s ratio ν :

$$\mu = \frac{E}{2(1+\nu)} \quad \lambda = \frac{E\nu}{(1+\nu)(1-2\nu)} \quad (2)$$

Plasticine Plasticine material is modeled with a combination of Saint Venant-Kirchhoff Model (StVK) and von Mises return mapping function. The energy density function of StVK can be formulated as

$$\Psi(\mathbf{F}) = \mu[\text{tr}(\mathbf{G}^2)] + \frac{\lambda}{2}[\text{tr}^2(\mathbf{G})] \quad (3)$$

where $\mathbf{G} = \frac{1}{2}(\mathbf{F}^\top \mathbf{F} - d)$ is the Green strain. The von-Mises return mapping function projects the deformation gradient back onto the boundary of the elastic region according to the von-Mises yielding condition. The mapping function can be formulated as

$$\mathcal{Z}(\mathbf{F}) = \begin{cases} \mathbf{F} & \delta\gamma \leq 0 \\ \mathbf{U} \exp(\epsilon - \delta\gamma \frac{\hat{\epsilon}}{\|\hat{\epsilon}\|}) \mathbf{V}^\top & \text{otherwise} \end{cases} \quad (4)$$

where $\mathbf{F} = \mathbf{U}\Sigma\mathbf{V}^\top$ is the singular value decomposition (SVD) of \mathbf{F} , $\epsilon = \log(\Sigma)$ is the Hencky strain, $\hat{\epsilon} = \epsilon - \bar{\epsilon}$ is the normalized Hencky strain and $\delta\gamma = \|\hat{\epsilon}\| - \frac{\tau_Y}{2\mu}$ is von-Mises yielding condition with the yield stress τ_Y as a physical parameter.

Sand Similar to the Plasticine material, we also use StVK as the constitutive model and apply its energy density function to the Sand material. The difference is that we use Drucker-Prager yield criteria instead of von-Mises yield criteria. The mapping function can be formulated as

$$\mathcal{Z}(\mathbf{F}) = \begin{cases} \mathbf{U}\mathbf{V}^\top & \text{tr}(\epsilon) > 0 \\ \mathbf{F} & \delta\gamma \leq 0, \text{tr}(\epsilon) \leq 0 \\ \mathbf{U} \exp(\epsilon - \delta\gamma \frac{\hat{\epsilon}}{\|\hat{\epsilon}\|}) \mathbf{V}^\top & \text{otherwise} \end{cases} \quad (5)$$

where $\delta\gamma = \|\hat{\epsilon}\|_F + \alpha \frac{(d\lambda+2\mu)\text{tr}(\epsilon)}{2\mu}$ is the yield stress, $\alpha = \sqrt{\frac{2}{3}} \frac{2\sin\theta_f}{3-\sin\theta_f}$ and θ_f is the friction angle.

3.2. Generalized to Complex Boundary Conditions

In this section, we demonstrate that the reconstruction results of our method, Vid2Sim, integrate seamlessly into the simulation of various animations under complex boundary conditions. Two examples are presented in Fig. 8, highlighting Vid2Sim’s ability to generate high-quality animations across diverse boundary scenarios.



Figure 3. More dynamic reconstruction results from the input videos.

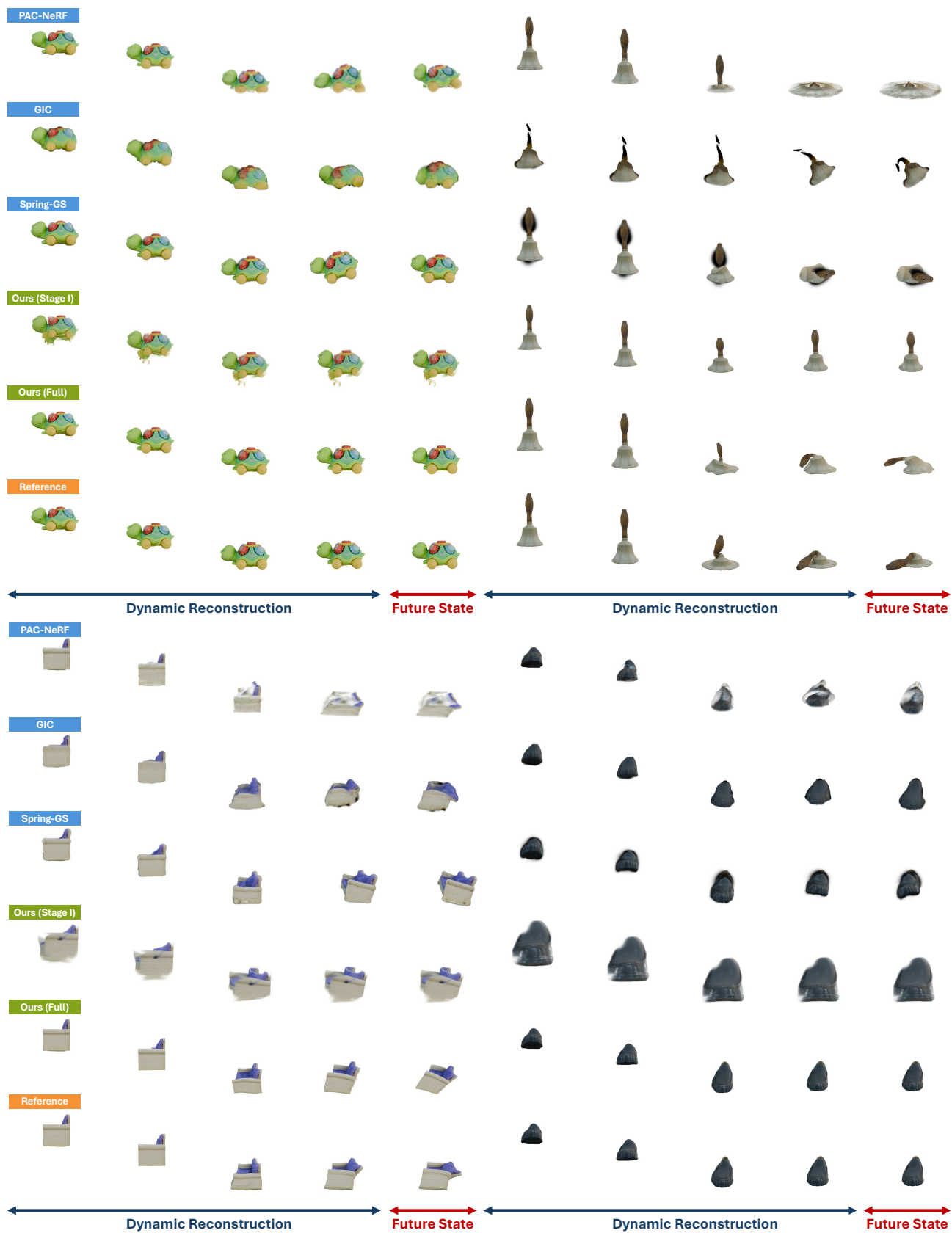


Figure 4. More dynamic reconstruction results from the input videos.

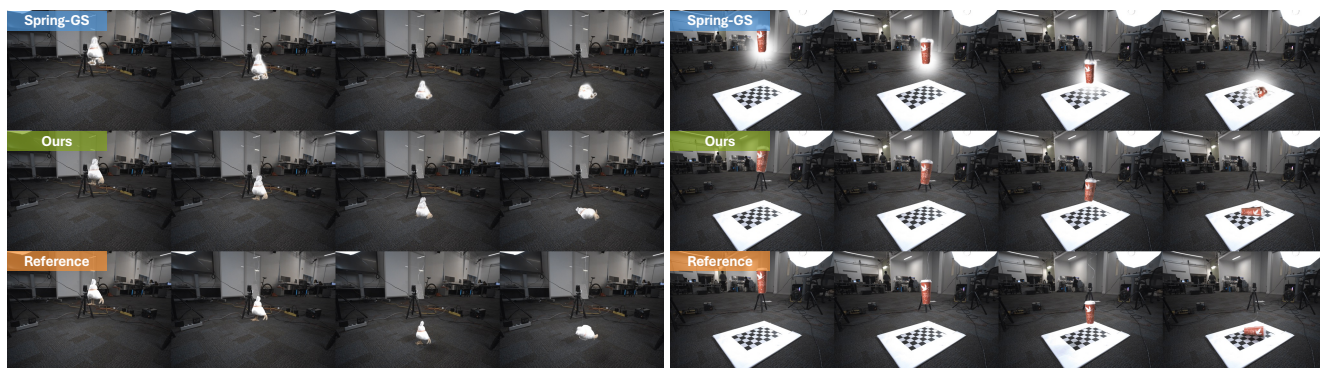


Figure 5. More dynamic reconstruction results from real-world input videos.



Figure 6. Novel view synthesis of the dynamic reconstruction results.

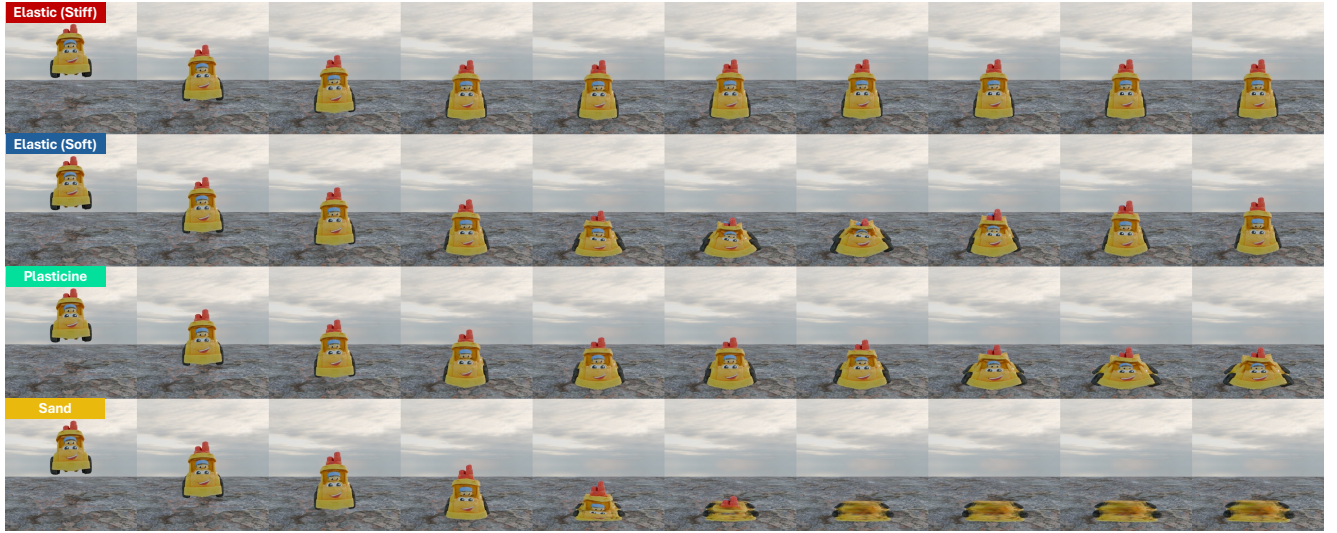
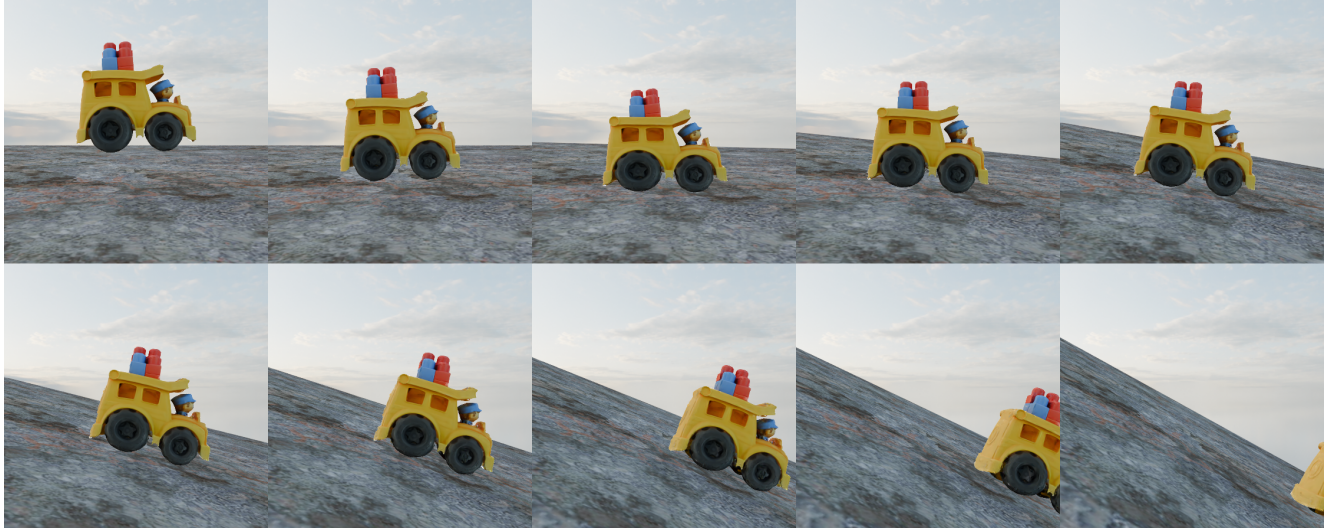
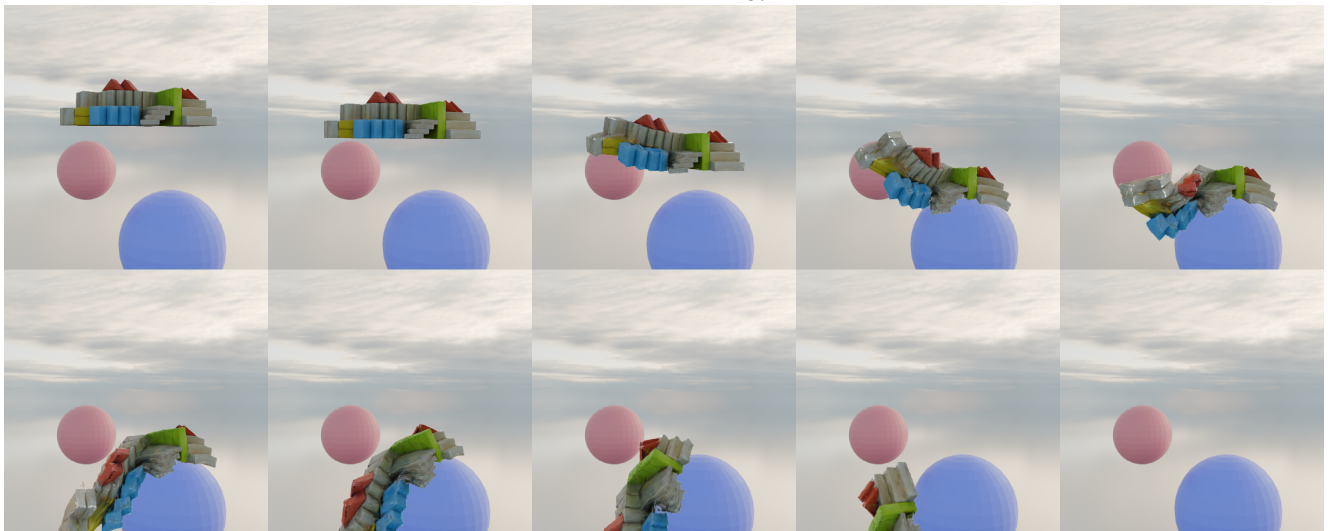


Figure 7. Simulation with different materials. We use $E = 10^7$, $\nu = 0.49$ for the stiff elastic and $E = 8000$, $\nu = 0.4$ for the soft elastic. In Plasticine material τ_γ is set to 500 and in Sand material θ_f is set to 10° .



(a) A bus slides at a *moving* floor.



(b) Blocks drop on the *balls*.

Figure 8. Simulation results based on different boundary conditions.

References

- [1] Noam Aigerman, Kunal Gupta, Vladimir G Kim, Siddhartha Chaudhuri, Jun Saito, and Thibault Groueix. Neural jacobian fields: Learning intrinsic mappings of arbitrary meshes. *arXiv preprint arXiv:2205.02904*, 2022. [1](#)
- [2] Junhao Cai, Yuji Yang, Weihao Yuan, Yisheng He, Zilong Dong, Liefeng Bo, Hui Cheng, and Qifeng Chen. Gaussian-informed continuum for physical property identification and simulation. *arXiv preprint arXiv:2406.14927*, 2024. [3](#)
- [3] Djork-Arné Clevert. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015. [1](#)
- [4] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. [1](#)
- [5] Xuan Li, Yi-Ling Qiao, Peter Yichen Chen, Krishna Murthy Jatavallabhula, Ming Lin, Chenfanfu Jiang, and Chuang Gan. Pac-nerf: Physics augmented continuum neural radiance fields for geometry-agnostic system identification. *arXiv preprint arXiv:2303.05512*, 2023. [3](#)
- [6] Vismay Modi, Nicholas Sharp, Or Perel, Shinjiro Sueda, and David IW Levin. Simplicits: Mesh-free, geometry-agnostic elastic simulation. *ACM Transactions on Graphics (TOG)*, 43(4):1–11, 2024. [1](#), [2](#)
- [7] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. Video-mae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *Advances in neural information processing systems*, 35:10078–10093, 2022. [1](#)
- [8] Licheng Zhong, Hong-Xing Yu, Jiajun Wu, and Yunzhu Li. Reconstruction and simulation of elastic objects with spring-mass 3d gaussians. In *European Conference on Computer Vision*, pages 407–423. Springer, 2025. [3](#)