LEARNING 3D-GAUSSIAN SIMULATORS FROM RGB VIDEOS

Anonymous authors
Paper under double-blind review

ABSTRACT

Realistic simulation is critical for applications ranging from robotics to animation. Video generation models have emerged as a way to capture real-world physics from data, but they often face challenges in maintaining spatial consistency and object permanence, relying on memory mechanisms to compensate. As a complementary direction, we present 3DGSim, a learned 3D simulator that directly learns physical interactions from multi-view RGB videos. 3DGSim adopts MVSplat to learn a latent particle-based representation of 3D scenes, a Point Transformer for the particle dynamics, a Temporal Merging module for consistent temporal aggregation, and Gaussian Splatting to produce novel view renderings. By jointly training inverse rendering and dynamics forecasting, 3DGSim embeds physical properties into point-wise latent features. This enables the model to capture diverse behaviors, from rigid and elastic to cloth-like dynamics and boundary conditions (e.g., fixed cloth corners), while producing realistic lighting effects. We show that 3DGSim can generate physically plausible results even in out-of-distribution cases, e.g. ground removal or multi-object interactions, despite being trained only on single-body collisions.

1 INTRODUCTION

Simulating visually and physically realistic environments is a cornerstone for embodied intelligence. Robots must soon tackle tasks such as opening washing machines, folding laundry, or tending plants. Traditional analytical simulators demand exact geometry, poses, and material parameters, making arbitrary scene simulation impractical. An alternative is to learn models that predict future states of a scene in large-scale observations, as evidenced by the striking visual realism of 2D video generation methods (Li et al., 2022; NVIDIA et al., 2025; Wu et al., 2023). However, pure 2D approaches lack 3D structure awareness, leading to failures in occlusion handling, object permanence, and physical plausibility (Motamed et al., 2025).

3D-based representations address many of these shortcomings, as shown by recent learned particle-based simulators (Allen et al., 2023; Li et al., 2019) which model a wide range of physical phenomena, from fluids and soft materials to articulated and rigid body dynamics. Yet, scaling such methods to data-rich regimes remains challenging, as most methods require privileged signals (object-level tracks, depth sensors, physics prior) or hand-crafted graph constructions.

To bridge this gap, we identify three pillars for generalizable, scalable visuo-physical simulation from videos: *(1) 3D visuo-physical reconstruction* from raw RGB observations; *(2) Imposing minimal physical biases* that can capture diverse physics; *(3) Efficient, differentiable decoding* back to image space for supervision via reconstruction loss.

Graph neural networks (GNNs) (Sanchez-Gonzalez et al., 2020; Shi et al., 2024; Wang et al., 2024; Whitney et al., 2023; 2024; Xue et al., 2023) have shown great promise in introducing relational inductive biases to handle the unstructured nature of particle sets. This has allowed GNN-based particle simulators to make major progress on all three pillars. In particular, Whitney et al. (2023) jointly train an encoder and dynamics model to learn visuo-physical pixel features from RGBD, and in the follow-up work Whitney et al. (2024) eliminate point correspondences via abstract temporal nodes or per-step models with merging. Driess et al. (2023) demonstrate end-to-end dynamics training of composable NeRF fields from raw RGB images. These advances, in combination with recent advances in feed-forward inverse rendering (Chen et al., 2024) and fast differentiable rendering of

Table 1: Overview on recently proposed particle-based simulators. While most works resort to a combination of kNN and GNNs, our work distinguishes itself by resorting to 3D Gaussian Splatting, space filling curves (SFC) for point cloud serialization, and training the inverse rendering encoder alongside a dynamics transformer.

Method (🔒: No data / code)		Scene representation	Inverse renderer (❄: Pretrained)	Graph synthesis	Dynamics model (🎖: Uses privileged info)	Forward rendering
SDF-Sim 🔒	Rubanova et al. Rubanova et al. (2024)	Mesh	n.a.	SDF	GNN 🎖	n.a.
PGNN.	Saleh et al. Saleh et al. (2024)	Mesh	n.a.	Mesh	GNN + Attention 🎖	n.a
FIGNet	Allen et al. Allen et al. (2022a)	Mesh faces	n.a.	BVH	GNN 🎖	n.a.
Robocraft	Shi et al. Shi et al. (2024)	Point clouds	n.a. (RGB-D)	kNN	GNN 🎖	NeRF
3DIntphys 🔒	Xue et al. Xue et al. (2023)	Point clouds	NeRF (Point sampl.) ❄	kNN	GNN 🎖	NeRF
VPD 🔒	Whitney et al. Whitney et al. (2023)	Point clouds	n.a. (RGB-D + UNet)	kNN	GNN	NeRF
HD-VPD 🔒	Whitney et al. Whitney et al. (2024)	Point clouds	n.a. (RGB-D + UNet)	kNN	GNN + Transformer	NeRF
DEL 🔒	Wang et al. Wang et al. (2024)	Point clouds	NeRF (GPF) ❄	kNN	GNN + DEM	NeRF
3DGSim	(Ours)	Gaussian splats	MVSplat	SFC	Transformer	3DGS

particles (Kerbl et al., 2023), encourage us to ask the question: *can we give up the inductive bias arising from locally connected graphs and still learn 3D particle-based simulators?*

To this end, we build 3DGSim, a fully end-to-end differentiable framework that embraces the power of scalable computation over hand-crafted biases. 3DGSim begins by inferring 3D visuo-physical features from raw multi-view RGB images through a feed-forward inverse renderer based on MVSplat. We then introduce a transformer-only dynamics engine, avoiding kNN-based graph construction and manually designed edge features in favor of learned spatiotemporal embeddings. Finally, a Gaussian Splatting head enables training on an image reconstruction loss from multi-view videos.

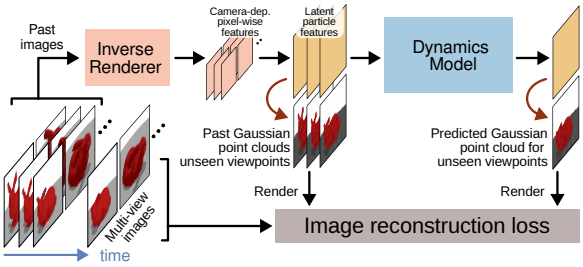

Figure 1: **3DGSim** works directly on **multi-view RGB videos** and is trained **end-to-end on next image prediction**. The **dynamics model** (transformer) operates on **particles with latent feature**. A learned mapping **transforms them into 3D Gaussian Splats** for novel view rendering.

Specifically, 3DGSim introduces the following key contributions:

- **Inverse Renderer**: Extends MVSplat with a feature extraction module fusing pixel-aligned features into a particle visuo-physical latent representation.
- **Temporal Encoding & Merging Layer**: Discards abstract temporal nodes in favor of a hierarchical module that processes an arbitrary number of timesteps.
- **Transformer-Only Dynamics Engine**: Removes graph biases and instead uses space-filling curves and learned embeddings for particle-based simulation.
- **End-to-End Differentiable Framework**: Connects inverse rendering, transformer dynamics, and Gaussian splatting-based decoding to training for next-frame image reconstruction.
- **Open Source Release**: We release the code and dataset to establish a reproducible baseline for future visuo-physical simulation research.

2 Related Work

Encoding and rendering scene representations Common 3D scene representations include point clouds (particles), meshes, signed distance functions (SDFs), neural radiance fields (NeRFs) (Mildenhall et al., 2021), and 3D Gaussians (splats) (Kerbl et al., 2023). Point clouds, which approximate object surfaces, can be obtained from RGB-D sensors (Shi et al., 2024; Whitney et al., 2023; 2024) or via inverse rendering (Chen et al., 2024; Murai et al., 2024; Wang et al., 2025). Works, such as Whitney et al. (2023; 2024), use U-Net–style encoders trained jointly with the dynamics model, allowing the extracted features to be optimized for physical prediction, a strategy shown to outperform independently trained encoders (Li et al., 2022). We adopt this joint training approach using MVSplat (Chen et al., 2024), where the encoded features are initially bound to camera parameters. To unify these visuo-physical latents in a global frame, we introduce a learned feature transformation module that maps them into a consistent 3D representation. While many PBS methods render from NeRFs (Driess et al., 2023; Shi et al., 2024; Wang et al., 2024; Whitney et al., 2023; 2024; Xue et al., 2023), we instead encode visual appearance directly in the particle cloud using

3D Gaussians. This explicit representation offers high rendering fidelity and significantly improved efficiency over NeRF-based rendering (Kerbl et al., 2023), supporting scalability.

GNN based particle-based simulators (PBS) Graph neural networks (GNNs) introduce relational inductive biases well-suited for modeling the unstructured nature of particle systems. Early work (Li et al., 2019; Sanchez-Gonzalez et al., 2020) demonstrated that GNN-based PBS can fit trajectories across a range of physical phenomena. However, GNNs struggle with rigid bodies, where instantaneous velocity changes require long-range message passing across the entire graph in a single step. To address this, later works incorporate mesh structures (Allen et al., 2022b; Pfaff et al., 2021) or signed distance functions (SDFs) (Rubanova et al., 2024) to enforce object-level coherence. Although effective in rigid-body settings, these methods do not generalize to deformable or fluid systems. Recent works (Saleh et al., 2024; Whitney et al., 2024) suggest adding attention layers to efficiently pass information through the graph. Wang et al. (2024) move toward greater data efficiency by incorporating physics-inspired biases such as the Material Point Method, though limiting broad applicability and requiring small simulation timesteps. To address temporal correspondence, Whitney et al. (2023) introduces abstract temporal nodes, while Whitney et al. (2024) combines GNNs with transformers to improve memory efficiency by processing and merging pairs of timesteps. However, the method is restricted to two-step horizons, as it requires training a separate model for each additional timestep. Methods based on GNNs rely on kNN to define point connectivities within a fixed radius and hand-crafted features based on object associations and distances to define graph features. Message passing and spatial pooling via furthest-point-sampling (FPS) are then used to aggregate information for dynamics prediction. However, kNN and distance computations are expensive and take up 54% of the forward time (Wu et al., 2024b), which limits scalability and prevents real-time forecasting. In contrast, we follow the design of PTv3 (Wu et al., 2024b). In 3DGSim, we trade off exact KNN neighborhood computation with space-filling curve–based ordering of particles and use sparse convolutions to encode relative positions, avoiding distance calculations. To enable the processing of temporal point clouds, we propose Temporal Merging with Grid Pooling to construct a hierarchical spatiotemporal, UNet-style Point Transformer for dynamics prediction.

Analytical particle simulators as physical prior Our work differs in purpose from applications which use Gaussian Splatting particles and analytical PBS as physical prior (e.g. off-the-shelf differentiable MPM simulator) to accomplish a series of tasks such as tracking (Abou-Chakra et al., 2024; Keetha et al., 2024; Luiten et al., 2024; Zhang et al., 2024a), dynamic scene reconstruction (Huang et al., 2023; Wu et al., 2024a; Yu et al., 2023), or animation (Lin et al., 2025; Xie et al., 2023; Zhang et al., 2024b). While analytical PBS can be used for parameter identification (Abou-Chakra et al., 2024), they are tailored to specific simulation scenarios. For a detailed comparison, refer to the supplementary material (see Appendix C.2).

3 Preliminaries

3DGSim is build atop several prior works, namely: 3D-Gaussian splatting which enables fast rendering, MVSplat which yields 3D Gaussian point clouds from multi-view images, and PTv3 which enables efficient neural processing of 3D point clouds.

Gaussian Splatting 3D Gaussian splatting (3DGS) (Kerbl et al., 2023) is an effective framework for multi-view 3D image reconstruction, representation, and fast image rendering and has gained rapid popularity due to its support for rapid inference, high fidelity, and editability of scenes. Gaussian splatting uses a collection of 3D Gaussian primitives, each parameterized by

$$g_i = (p_i, c_i, r_i, s_i, \sigma_i), \tag{1}$$

with the Gaussian's mean p_i (particle position), its rotation r_i, spherical harmonics c_i (defines coloring), scale s_i, and opacity σ_i. To render novel views, these primitives are projected onto a 2D image plane using differential tile-based rasterization. The color value at pixel $\mathbf{p}$ is calculated via alpha-blend rendering: $I(\mathbf{p}) = \sum_{i=1}^{N} \alpha_i c_i \prod_{j=1}^{i-1}(1-\alpha_j)$ where $\alpha_i = \sigma_i e^{-\frac{1}{2}(\mathbf{p}-p_i)^\top \Sigma_i^{-1}(\mathbf{p}-p_i)}$ is the 2D density, I is the rendered image, N is the number of primitives in the image and Σ_i is the covariance matrix given by $\Sigma_i = r_i s_i r_i^\top$ for improved computational stability.

MVSplat: Multi-view feed-forward 3D reconstruction MVSplat deploys a feed-forward network f_ϕ with parameters ϕ that maps M images $\mathcal{I} = \{I^m\}_{i=m}^M$ with $I^m \in \mathbb{R}^{(H\times W\times 3)}$ to a set of pixel-aligned 3D Gaussian primitives (Fig. 2)

$$f_\phi : \{(I^m, P^m)\}_{m=1}^M \mapsto \{g_i\}_{i=1}^{M\times H\times W}.$$

At each time step, MVSplat localizes Gaussian centers using a cost volume representation through plane-sweeping and cross-view feature similarities. To do so, it requires the corresponding camera projection matrices $\mathcal{P} = \{P^m\}_{m=1}^M$ that are calculated as $P^m = K^m[R^m|t^m]$ via the camera intrinsics K^m, rotation R^m, and translation t^m.

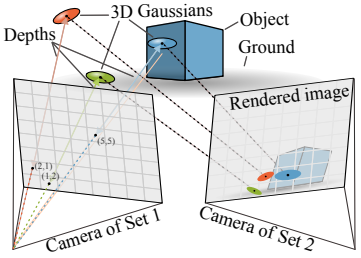

Figure 2: MVSplat uses a cost volume with plane sweeping to regress pixel-wise 3D Gaussians, which are unprojected to world frame using camera parameters.

4 3DGSIM

3DGSim is a fully differentiable pipeline that, given T past multi-view RGB frames, reconstructs 3D particles with latent features, simulates their motion, and renders the next frames. It consists of three jointly trained modules (Fig.1): (i) an *encoder* that maps multi-view RGB images to 3D particles, (ii) a *dynamics model* that simulates the motion of these particles through time, and (iii) a *renderer* that yields images by first mapping the particles to Gaussian splats.

4.1 STATE REPRESENTATION

To simulate physical scenes from vision, we require a state representation that is both expressive enough to capture fine-grained 3D and physical properties, and compact enough to enable efficient learning and prediction. Although an explicit 3DGS representation $g_i(t_k)$ offers geometric and visual completeness, it is insufficient for dynamics modeling. Instead, we distill the state of each particle into a more compact representation:

$$\tilde{g}_i(t_k) = \big(p_i(t_k),\ f_i(t_k)\big) \tag{2}$$

where t_k denotes the k-th timestep and $f_i \in \mathbb{R}^d$ the visuo-physical latent particle feature, encoding shape, appearance, and dynamic properties. Unless otherwise stated, we omit the timestep t_k and the particle index i when the statement applies to all timesteps or particles, respectively.

Optional: Masking and Freezing of Particles At each timestep t_k, the encoder yields pixel-aligned features for each input image. As an optional step, one can apply a foreground mask to discard particles likely belonging to the static background, retaining a reduced set of N_k particles per time step (Fig. 2). Additionally, as originally suggested by Whitney et al. (2023), static particles can optionally be "frozen", i.e. act as input to the dynamics model but are excluded from position updates. These optional strategies improve efficiency without being necessary for successful training, as shown in Section 5 and Appendix B.

Invariant and dynamic feature decomposition We decompose each particle's visuo-physical feature into an invariant and a dynamic part as shown in Fig. 3, writing

$$f_i = f_i^{\text{inv}} \oplus f_i^{\text{dyn}},$$

where $\oplus$ denotes concatenation. The dynamics model updates only f_i^{dyn}, while leaving f_i^{inv} unchanged. For clarity, we will refer to the dynamics update as "updating f_i", though only the dynamic component f_i^{dyn} is altered.

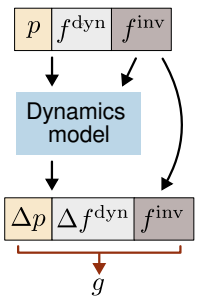

Figure 3: Position p and dynamic features f^{dyn} are updated while f^{inv} remain constant.

4.2 View-Independent Inverse Renderer

In MVSplat, pixel-aligned features $\hat{f}_i'$ are tied to the specific camera view from which they were extracted. While Gaussian primitives (e.g. depth, scale, rotation, harmonics) can be directly unprojected or transformed into the world frame using camera parameters, latent features remain bound to the camera-centric frame. Since dynamics predictions are invariant to the observer's viewpoint, such a dependence on view-dependent encodings hampers generalization.

To overcome this, 3DGSim introduces a feature encoding network that maps pixel-aligned features $\hat{f}_i'$ into view-independent latent representations f_i. The encoder employs FiLM conditioning (Perez et al., 2017) on pixel depth, pixel shift, density, and ray geometry (parameterized via Plücker coordinates (Plücker, 1868-1869)) to infer spatially consistent 3D features. As a result, the inverse rendering module produces canonically anchored particle states, providing a unified representation for downstream dynamics learning. Further architectural details are described in Appendix A.1.

4.3 Dynamics Model

At the core of our method is the dynamics model, a transformer architecture operating on particle sets in space and time. The dynamics model receives as input T past particle sets,

$$\left\{\{\tilde{g}_i(t_k)\}_{i=1}^{N_k}\right\}_{k=1}^{T}, \quad \text{where } \tilde{g}_i(t_k) = \left(p_i(t_k), f_i^{\text{inv}}(t_k), f_i^{\text{dyn}}(t_k)\right), \tag{3}$$

and predicts the updated dynamic features at the next timestep

$$\Delta p(t_T), \Delta f^{\text{dyn}}(t_T) = \text{Dynamics Model}\left(\left\{\{\tilde{g}_i(t_k)\}_{i=1}^{N_k}\right\}_{k=1}^{T}\right), \tag{4}$$

such that $p_i(t_{T+1}) = p_i(t_T) + \Delta p_i(t_T)$ and $f_i^{\text{dyn}}(t_{T+1}) = f_i^{\text{dyn}}(t_T) + \Delta f_i^{\text{dyn}}(t_T)$. As these point clouds are unstructured and potentially vary in size at each time step due to masking, a fundamental challenge arises: *How can a network efficiently propagate the embedded physics information both spatially and temporally?*

We tackle this question by building on PTv3 Wu et al. (2024b), which has recently achieved state-of-the-art performance in representation learning for unstructured point clouds Wu et al. (2025). As discussed in Appendix A.2, PTv3 operates by serializing the input point cloud and applying patch-wise attention. However, the original design of PTv3 is limited to point clouds that do not exhibit temporal variation. In this section, we extend PTv3 to predict dynamics from *temporally evolving point clouds*. First, we extend serialization to equip point cloud encodings with a timestamp. Then, we equip features with temporal embeddings that allow attention to distinguish timestamps. Lastly, we use the timestamps to merge neighboring latent particle sets, enabling PTv3's patch-wise attention blocks to aggregate information across time.

Temporally serialized point cloud (t-SPC) To enable spatio-temporal reasoning over multiple timesteps, we extend PTv3's point serialization scheme by encoding both spatial and temporal structure into a single key. Specifically, for each particle i at timestep t_k in batch b, we define a 64-bit serialization code:

$$\tilde{s}_i(t_k, b) = \Big[\underbrace{b}_{(64\text{ - }\tau\text{ - }\kappa)\text{ Bits}} \Big| \underbrace{s_{t_k}}_{\tau\text{ Bits}} \Big| \underbrace{s_i}_{\kappa\text{ Bits}} \Big] \tag{5}$$

Here, s_{t_k} is the temporal code and s_i is a spatial code obtained by projecting p_i onto a space-filling curve (SFC). We set $\kappa = 48$ and allocate $\tau = \log_2(T)$ bits for time. With 16 bits per dimension and a grid resolution of $G = 0.004\,\text{m}$, the spatial encoding spans up to $216\,\text{m}$ per axis.

Temporal encoding As shown in Fig. 6, before merging t-SPCs across timesteps, we inject a learned, timestep-specific positional encoding E_{t_k} as

$$f_i(t_k) \leftarrow f_i(t_k) + E_{t_k}. \tag{6}$$

This temporal encoding ensures that the attention mechanism can distinguish points across different temporal instances, enabling the model to reason about dynamics over time. Similar positional encoding methods have previously been applied in transformer architectures to differentiate positions within sequences (Vaswani et al., 2017).

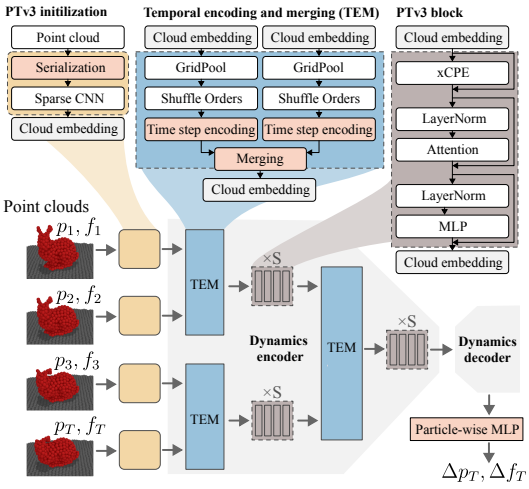

Figure 4: **The dynamics model** encodes the time step into each embedding and merges embeddings from adjacent timesteps. The TEM and PTv3 blocks are applied repeatedly until all embeddings are merged. Our extensions to PTv3 are highlighted in red.

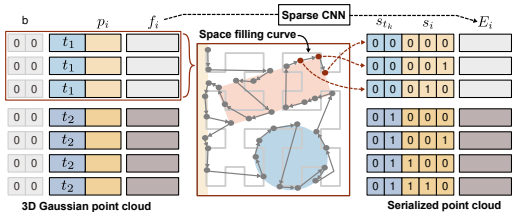

Figure 5: Spatio-temporal point cloud serialization.

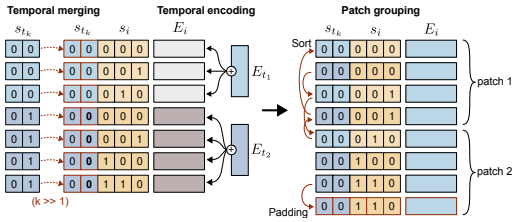

Figure 6: Temporal merging and embedding followed by patch grouping for applying patch-wise attention.

Temporal merging Unlike PTv3, which restricts attention exclusively to patches composed of points from the *same time step*, our method enables a wider receptive field across time. To do so, we propose *temporal merging* which applies a one-bit right shift to the temporal codes s_{t_k}:

$$\text{Merge}(\tilde{s}_i) = [b \mid (s_{t_k} \gg 1) \mid s_i]. \tag{7}$$

For instance, points from time steps $s_{t_1} = 0$ and $s_{t_2} = 1$ are merged by shifting their codes, so they both become 0, as depicted in Fig. 6. By grouping points from separate time steps into a single patch, the attention module can model relationships across time.

Importantly, while Whitney et al. (2024) deploy a dedicated transformer module for each time step, our proposition of temporal merging enables the reuse of the same attention block across time steps, which significantly reduces memory consumption and promotes knowledge transfer.

Patch-wise attention and particle-wise MLP After each temporal encoding and merging (TEM) block, the cloud embeddings are processed by PTv3's patch-wise attention block. First, the embeddings are equipped with a position encoding via a sparseCNN with skip connection (xCPE in Fig. 4). Then, the embeddings are fed to a patch-wise attention layer. Finally, at the end of the dynamics model, each particle alongside its embedding is mapped by a particle-wise MLP to Δp_T and Δf_T.

4.4 Rendering Features for the Image Reconstruction Loss

To render images with 3DGS, particle states $\tilde{g}_i = (p_i, f_i)$ are transformed into Gaussian splat parameters g_i via a learned head, materialized only at the final stage to supervise the training with image reconstruction.

3DGSim is trained solely on an image reconstruction loss $\mathcal{L}$. This loss is computed from rasterized multi-view images, generated based on both the encoder predictions of past point clouds $\{\{g_i(t_k)\}_{i=1}^{N_k}\}_{k=0}^{T}$ and the simulated future point cloud trajectory $\{\{g_i(t_k)\}_{i=1}^{N_k}\}_{k=T+1}^{T+T'}$. Specifically, the loss reads

$$\mathcal{L} = (1-\lambda)\frac{1}{T}\sum_{k=0}^{T}\mathcal{L}_k + \lambda \sum_{k=T+1}^{T+T'} \gamma^{k-T-1}\mathcal{L}_k \quad \text{and} \quad \mathcal{L}_k = \mathcal{L}_2(I_k^{\text{gt}}, I_k) + \beta\,\mathcal{L}_{\text{LPIPS}}(I_k^{\text{gt}}, I_k), \tag{8}$$

with $\lambda = 0.5$, temporal decay factor $\gamma = 0.87$, $T \in \{2, 4\}$ and $T' = 12$. The per-frame reconstruction loss $\mathcal{L}_k$ measures the discrepancy between ground-truth (I_k^{gt}) and predicted (I_k) multi-view images using a weighted combination of pixel-wise ℓ_2 and LPIPS Zhang et al. (2018) terms with hyperparameter $\beta = 0.05$.

5 EXPERIMENTS

In what follows, we train 3DGSim on different datasets and test the model's ability to generalize.

Model setup Unless stated otherwise, the following training and parameter settings serve as defaults in the experiments. The state consists of dynamic f^{dyn} and invariant features f^{inv} of size $(32, 32)$ for the implicit- and $(n_f, 16)$ for the explicit 3D Gaussian particle representation. In the explicit representation, f^{dyn} corresponds to explicit Gaussian primitives of size n_f which are directly used for rendering. The inverse rendering encoder follows MVSplat, reducing candidate depths from 128 to 64 due to smaller scene distances. Default near-far depth ranges are $[0.2, 4]$ for rigid bodies and $[1.5, 8]$ for the other datasets, as the scene has a larger scale. The dynamics transformer defaults to PTv3 with a 5-stage encoder (block depths $[2, 2, 2, 6, 2]$) and a 4-stage decoder ($[2, 2, 2, 2]$). Grid pooling and temporal merging strides default to $[1, 4, 2, 2, 2]$ and $[1, 2, 2, 2, 2]$, respectively, with grid size G=0.004 m. Attention blocks use patches of size 1024, encoder feature dimensions $[32, 64, 128, 256, 512]$, decoder dimensions $[64, 128, 256]$, encoder heads $[2, 4, 8, 16, 32]$, and decoder heads $[4, 4, 8, 16]$. For the camera setup, we select 4 uniformly distributed views at random and an additional 5 target cameras from the remaining cameras (out of 12 total) to compute the reconstruction loss.

Training Our models are trained with AdamW for ∼120,000 steps using a cosine annealing warm-up and a learning rate of 2×10^{-4}, with batch sizes of 6 and 4 for 2-step and 4-step states, respectively. To optimize memory and speed, we use gradient checkpointing and flash attention v2 (Dao, 2024). Training is performed on a single H100 GPU and typically takes around six days.

Datasets To evaluate 3DGSim's robustness in learning dynamics from videos, we introduce three challenging datasets: rigid body, elastic, and cloth.

The rigid body dataset consists of 1,000 simulated trajectories from the MOVI dataset, involving six rigid objects (turtle, sonny school bus, squirrel, basket, lacing sheep, and turboprop airplane) from the GSO dataset (Downs et al., 2022). Each trajectory spans 32 frames at 12 FPS, providing controlled dynamics characteristic of rigid body motion. The elastic dataset, aimed at capturing plastic deformable object dynamics, includes six objects (dragon, duck, kawaii demon, pig, spot, and worm) simulated using the Genesis MPM elastoplastic simulator (Authors, 2024). Each object undergoes deformation upon collision with a circular gray ground, offering scenarios of complex elastic behavior. The cloth dataset includes the same set of objects as the elastic dataset. Here, the cloth is fixed at four corners, posing the challenge to infer implicit constraints and modeling dynamic cloth-like deformations.

Both elastic and cloth datasets include 200 trajectories per object, simulated with a 0.001 time step and 20 substeps. Each two second sequence is recorded at 42 FPS resulting in 84 frames per trajectory and less than 6 minutes of footage per object.

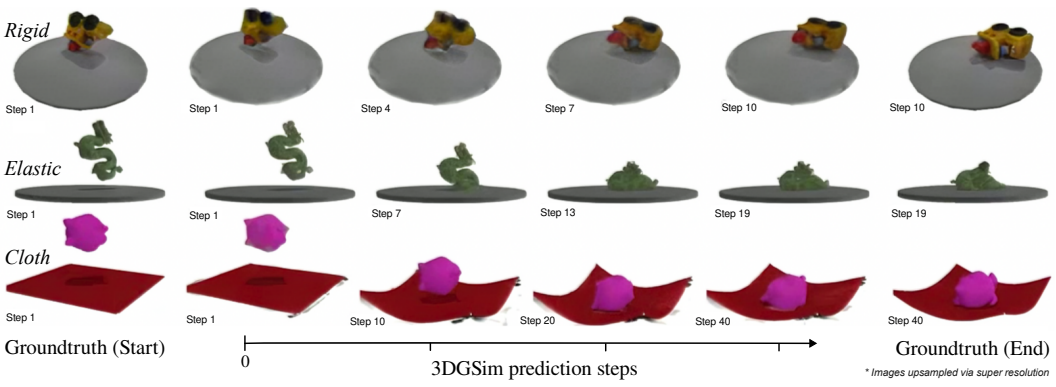

Figure 7: Qualitative examples of 3DGSim's dynamic predictions. After training on less than 6 minutes of video per object across 6 objects, 3DGSim accurately predicts motion of elasto-plastic deformations, rigid bodies, cloth.

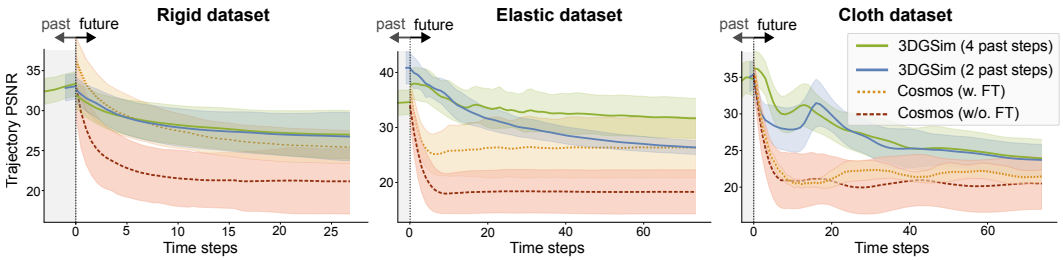

Figure 8: Trajectory PSNR of 3DGSim, Cosmos and CosmosFT is shown for both past and future predictions. The Cosmos models are conditioned on past frames and appropriate language prompts.

Figure 9: 3DGSim's prediction of a rigid plane captures shadows by altering ground particle appearance.

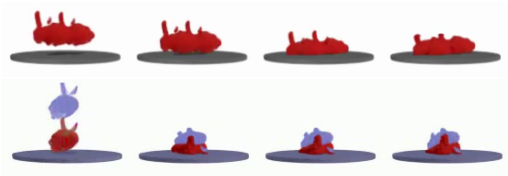

Figure 10: Although not trained on this specific elastic object or multiple objects, 3DGSim predicts physically plausible deformations.

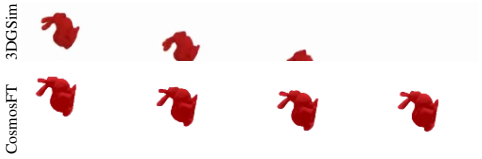

Figure 11: When the ground is removed, 3DGSim predicts the freefall, while CosmosFT hallucinates a levitating object at ground level.

Figure 12: Comparison of 3DGSim to Cosmos. *Explicit* models use 3DGS parameters and a static latent feature as inputs to the dynamics model, while *latent* models use only latent features mapped to Gaussians after dynamics. The †-model uses only 6 camera views (3 input + 3 reconstruction) instead of 12 (4+5); the ‡-model omits segmentation masks for static elements. Metrics *future* and *past* are means over all timesteps. "4-12" means 4 past steps predicting 12 future steps.

Dataset	Model	PSNR (future) ↑	PSNR (past) ↑	SSIM ↑	LPIPS ↓
Rigid	3DGSim 4-12 latent	**28.28 ± 2.52**	32.93 ± 1.56	**0.90 ± 0.03**	**0.09 ± 0.03**
	3DGSim 2-12 latent	**28.08 ± 2.46**	33.00 ± 1.62	**0.90 ± 0.03**	**0.09 ± 0.03**
	3DGSim 4-12 explicit	27.88 ± 2.43	32.77 ± 1.57	0.90 ± 0.03	0.09 ± 0.03
	3DGSim 2-12 explicit	27.07 ± 2.27	32.67 ± 1.65	0.90 ± 0.03	0.09 ± 0.03
	CosmosFT	26.44 ± 2.26	–	0.68 ± 0.05	0.10 ± 0.03
	Cosmos	22.35 ± 3.82	–	0.83 ± 0.08	0.24 ± 0.08
Elastic	3DGSim 4-12 latent	**33.15 ± 3.51**	34.55 ± 2.26	**0.97 ± 0.02**	**0.02 ± 0.01**
	3DGSim 2-12 latent	**32.05 ± 3.48**	35.99 ± 1.88	**0.96 ± 0.02**	**0.03 ± 0.02**
	3DGSim 2-12 explicit	29.92 ± 1.72	40.85 ± 2.94	0.96 ± 0.02	0.03 ± 0.02
	3DGSim 4-12 explicit	29.69 ± 1.75	40.16 ± 3.07	0.97 ± 0.02	0.02 ± 0.01
	3DGSim 4-12 latent †	**31.60 ± 3.09**	32.55 ± 2.12	**0.97 ± 0.02**	**0.02 ± 0.01**
	3DGSim 4-12 latent ‡	**32.66 ± 3.43**	34.45 ± 2.44	**0.96 ± 0.02**	**0.03 ± 0.02**
	CosmosFT	26.50 ± 5.21	–	0.82 ± 0.02	0.07 ± 0.03
	Cosmos	18.87 ± 3.99	–	0.79 ± 0.08	0.23 ± 0.08
Cloth	3DGSim 4-8 latent	**26.98 ± 2.63**	34.81 ± 2.28	**0.89 ± 0.03**	**0.08 ± 0.03**
	3DGSim 2-8 latent	**26.25 ± 2.38**	35.22 ± 1.97	**0.88 ± 0.03**	**0.08 ± 0.02**
	3DGSim 4-8 explicit	23.72 ± 1.52	39.75 ± 2.32	0.89 ± 0.03	0.08 ± 0.03
	3DGSim 2-8 explicit	17.97 ± 2.02	35.47 ± 1.68	0.88 ± 0.03	0.08 ± 0.02
	CosmosFT	22.49 ± 0.99	–	0.73 ± 0.03	0.14 ± 0.04
	Cosmos	21.10 ± 3.56	–	0.86 ± 0.06	0.19 ± 0.06

Figure 13: CosmosFT merges distinct worms into one before ground contact, even on in-distribution cases.

Benchmarking Existing 3D baselines do not allow direct comparison without substantial reimplementation. Key methods – VPD, HD-VPD, DEL, and 3D-IntPhys (Wang et al., 2024; Whitney et al., 2023; 2024; Xue et al., 2023) – *lack public code and data*, also unavailable upon contacting authors. Without published datasets, any reimplementation would lack verifiability, limiting reproducibility and fair evaluation. To address this, we will release our code and datasets. DPI-Net and VGPLDP (Li et al., 2019; 2020) are open-source but rely on ground-truth particle trajectories and require major adaptation to fit our setting. For 2D baselines, we provide quantitative comparison to Cosmos (NVIDIA et al., 2025). Note that Cosmos differs from our multi-view setup as it is pretrained on multiple past frames from a single view. For fair comparison, we evaluated both the base model and a LoRA-finetuned variant of Cosmos-Predict2 (CosmosFT) trained for 6,000 iterations on our data set using recommended parameters. The Cosmos models are conditioned on the prompts detailed in Table S7). For evaluation, 12% of trajectories are chosen at random and held out from each dataset, and we report each model's PSNR, LPIPS and SSIM.

5.1 Trajectory Simulation

3DGSim achieves competitive long-horizon simulation accuracy; up to 80 steps; compared with state-of-the-art baselines such as Cosmos-1.0-Autoregressive-5B-Video2World and a LoRA-finetuned Cosmos-Predict2 (NVIDIA et al., 2025). Performance curves are shown in Fig. 8. Ablation studies (Tab. 12) reveal that keeping 3DGS primitives explicit in the representation yields similar short-term performance but generalizes poorly, especially with fewer cameras (see Appendix B). By contrast, using a latent implicit representation leads to more robust generalization.

5.2 Scene Editing and Model Generalization

With its explicit 3D state, 3DGSim supports direct scene editing, providing a natural testbed for generalization. When the ground is raised or removed, conditions never seen during training, the model continues to generate stable, physically consistent rollouts (Fig. 11). This suggests a robust grasp of underlying dynamics that extends beyond the training distribution.

We further test generalization by duplicating objects and running long-horizon simulations (Fig. 10), Appendix D.1). Although trained only on single object–ground collisions, 3DGSim accurately captures realistic multi-body interactions, with objects retaining integrity rather than collapsing into chaotic overlaps. Beyond interactions, it even models emergent properties such as shadows (Fig. 9), indicating a holistic understanding of lighting and geometry alongside physics.

In contrast, CosmosFT struggles under similar 2D-edits. When the ground is removed, objects often remain suspended (Fig. 11), and when multiple objects are introduced, they morph into a single mass before contact (Fig. 13). These hallucinations reflect the limits of 2D image-based reasoning, underscoring the advantages of an explicit 3D representation for robust and interpretable generalization. Further examples are shown in the supplementary.

5.3 Simulation Speed

Simulation speed is critical for robotics applications. Traditional simulators (FEM, MPM, PBD) typically employ small integration timesteps. Learned approaches enable larger timesteps, allowing 3DGSim to simulate elastic cloth at 42 FPS and rigid dynamics at 12 FPS, with inference speeds of $\sim$16 FPS (4 past steps) and $\sim$20.1 FPS (2 past steps), using under 20 GB VRAM on an H100 GPU and achieving *near real-time speeds* as illustrated in Fig. 14.

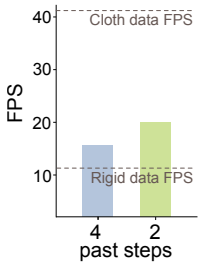

Figure 14: Prediction speed of 3DGSim versus simulation FPS.

6 Discussion

We introduced 3DGSim, a fully differentiable 3D Gaussian simulator that learns directly from multi-view RGB video. 3DGSim integrates inverse rendering, dynamics prediction, and novel-view video synthesis within a single end-to-end learnable system. Given that 3DGSim pioneers an unexplored direction for 3D particle-based simulation, future work will explore action conditioning, a natural next step that provides essential supervision signals for forecasting. This will also enable large-scale validation on real-world multi-view datasets, which are currently unavailable for passive phenomena. Our dependence on multi-view inputs could be further mitigated by recent advances in monocular inverse rendering (Murai et al., 2024; Wang et al., 2025). Additionally, while occlusions are not explicitly modeled, they are partially addressed by the dynamics module and may be further improved through point completion techniques.

Spatial Causality In 3DGSim, interactions are restricted to those between spatially grounded particles, which ensures that the simulation adheres to realistic physical dynamics. This contrasts with 2D pixel-based video generation models, where apparent dynamics often emerge from the generative flexibility of image-space synthesis. The 3D formulation thus brings advantages such as spatial consistency, object permanence, and robustness to out-of-distribution inputs, as exemplified by our generalization tests. However, it introduces certain compromises: while 2D predictors can effortlessly repurpose pixels to synthesize novel content, e.g., fabricating unseen objects from generative priors, 3D particle models inherit stricter structural constraints, making it difficult to dynamically create or destroy particles in a learnable manner. This highlights a tradeoff between the stability and interpretability provided by 3D spatial causality and the generative freedom unlocked by 2D video models.

Toward Vision-Language Simulation Integrating language embeddings offers a promising avenue for enriching particle-based simulations. Terms such as “liquid” or “mirror” provide informative priors about object properties, enabling more structured and semantically aware predictions. We envision 3DGSim as a step toward scalable simulators that can learn physical interactions from both visual and textual modalities, ultimately supporting a more nuanced robotic understanding of complex real-world dynamics.

REFERENCES

Jad Abou-Chakra, Krishan Rana, Feras Dayoub, and Niko Sünderhauf. Physically Embodied Gaussian Splatting: A Realtime Correctable World Model for Robotics, 2024.

Kelsey R. Allen, Yulia Rubanova, Tatiana Lopez-Guevara, William Whitney, Alvaro Sanchez-Gonzalez, Peter Battaglia, and Tobias Pfaff. Learning rigid dynamics with face interaction graph networks, 2022a.

Kelsey R. Allen, Yulia Rubanova, Tatiana Lopez-Guevara, William Whitney, Alvaro Sanchez-Gonzalez, Peter Battaglia, and Tobias Pfaff. Learning rigid dynamics with face interaction graph networks, 2022b.

Kelsey R Allen, Tatiana Lopez Guevara, Yulia Rubanova, Kim Stachenfeld, Alvaro Sanchez-Gonzalez, Peter Battaglia, and Tobias Pfaff. Graph network simulators can learn discontinuous, rigid contact dynamics. In *Conference on Robot Learning*, pp. 1157–1167. PMLR, 2023.

Genesis Authors. Genesis: A universal and generative physics engine for robotics and beyond, 2024.

Jeongmin Bae, Seoha Kim, Youngsik Yun, Hahyun Lee, Gun Bang, and Youngjung Uh. Per-gaussian embedding-based deformation for deformable 3d gaussian splatting. In *European Conference on Computer Vision (ECCV)*, 2024.

Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. *arXiv preprint arXiv:2403.14627*, 2024.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.

Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B. McHugh, and Vincent Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items, 2022.

Danny Driess, Zhiao Huang, Yunzhu Li, Russ Tedrake, and Marc Toussaint. Learning multi-object dynamics with compositional neural radiance fields. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 1755–1768, 2023.

Yi-Hua Huang, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. Sc-gs: Sparse-controlled gaussian splatting for editable dynamic scenes. *arXiv preprint arXiv:2312.14937*, 2023.

Nikhil Keetha, Jay Karhade, Krishna Murthy Jatavallabhula, Gengshan Yang, Sebastian Scherer, Deva Ramanan, and Jonathon Luiten. Splatam: Splat, track and map 3d gaussians for dense rgb-d slam. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023.

Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *ICLR*, 2019.

Yunzhu Li, Toru Lin, Kexin Yi, Daniel Bear, Daniel L.K. Yamins, Jiajun Wu, Joshua B. Tenenbaum, and Antonio Torralba. Visual grounding of learned physical models. In *International Conference on Machine Learning*, 2020.

Yunzhu Li, Shuang Li, Vincent Sitzmann, Pulkit Agrawal, and Antonio Torralba. 3d neural scene representations for visuomotor control. In *Conference on Robot Learning*, pp. 112–123. PMLR, 2022.

Yuchen Lin, Chenguo Lin, Jianjin Xu, and Yadong Mu. OmniPhysGS: 3D Constitutive Gaussians for General Physics-Based Dynamics Generation, January 2025. URL http://arxiv.org/abs/2501.18982. arXiv:2501.18982 [cs].

Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *3DV*, 2024.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

Saman Motamed, Laura Culp, Kevin Swersky, Priyank Jaini, and Robert Geirhos. Do generative video models understand physical principles? *arXiv preprint arXiv:2501.09038*, 2025.

Riku Murai, Eric Dexheimer, and Andrew J. Davison. MASt3R-SLAM: Real-Time Dense SLAM with 3D Reconstruction Priors, December 2024. URL http://arxiv.org/abs/2412.12392. arXiv:2412.12392 [cs].

NVIDIA, :, Niket Agarwal, Arslan Ali, Maciej Bala, et al. Cosmos world foundation model platform for physical ai, 2025.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. Film: Visual reasoning with a general conditioning layer. *CoRR*, 2017.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021.

Julius Plücker. *Neue Geometrie des Raumes gegründet auf die Betrachtung der geraden Linie als Raumelement*. Teubner, Leipzig, 1868-1869.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pp. 234–241. Springer, 2015.

Yulia Rubanova, Tatiana Lopez-Guevara, Kelsey R. Allen, William F. Whitney, Kimberly Stachenfeld, and Tobias Pfaff. Learning rigid-body simulators over implicit shapes for large-scale scenes and vision, 2024.

Mahdi Saleh, Michael Sommersperger, Nassir Navab, and Federico Tombari. Physics-encoded graph neural networks for deformation prediction under contact. *arXiv preprint arXiv:2402.03466*, 2024.

Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. Learning to Simulate Complex Physics with Graph Networks, 2020.

Haochen Shi, Huazhe Xu, Zhiao Huang, Yunzhu Li, and Jiajun Wu. Robocraft: Learning to see, simulate, and shape elasto-plastic objects in 3d with graph networks. *The International Journal of Robotics Research*, 43(4):533–549, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. VGGT: Visual Geometry Grounded Transformer, March 2025. URL http://arxiv.org/abs/2503.11651. arXiv:2503.11651 [cs]version: 1.

Jiaxu Wang, Jingkai Sun, Junhao He, Ziyi Zhang, Qiang Zhang, Mingyuan Sun, and Renjing Xu. DEL: Discrete Element Learner for Learning 3D Particle Dynamics with Neural Rendering, 2024.

William F. Whitney, Tatiana Lopez-Guevara, Tobias Pfaff, Yulia Rubanova, Thomas Kipf, Kimberly Stachenfeld, and Kelsey R. Allen. Learning 3d particle-based simulators from rgb-d videos, 2023.

William F. Whitney, Jacob Varley, Deepali Jain, Krzysztof Choromanski, Sumeet Singh, and Vikas Sindhwani. Modeling the real world with high-density visual particle dynamics, 2024.

Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20310–20320, 2024a.

Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer v3: Simpler, faster, stronger. In *CVPR*, 2024b.

Xiaoyang Wu, Daniel DeTone, Duncan Frost, Tianwei Shen, Chris Xie, Nan Yang, Jakob Engel, Richard Newcombe, Hengshuang Zhao, and Julian Straub. Sonata: Self-Supervised Learning of Reliable Point Representations, March 2025. URL http://arxiv.org/abs/2503.16429. arXiv:2503.16429 [cs].

Ziyi Wu, Nikita Dvornik, Klaus Greff, Thomas Kipf, and Animesh Garg. Slotformer: Unsupervised visual dynamics simulation with object-centric models, 2023.

Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. *arXiv preprint arXiv:2311.12198*, 2023.

Haotian Xue, Antonio Torralba, Josh Tenenbaum, Dan Yamins, Yunzhu Li, and Hsiao-Yu Tung. 3d-intphys: Towards more generalized 3d-grounded visual intuitive physics under challenging scenes. *Advances in Neural Information Processing Systems*, 36:7116–7136, 2023.

Heng Yu, Joel Julin, Zoltán Á Milacski, Koichiro Niinuma, and László A. Jeni. CoGS: Controllable Gaussian Splatting, 2023.

Mingtong Zhang, Kaifeng Zhang, and Yunzhu Li. Dynamic 3d gaussian tracking for graph-based neural dynamics modeling. In *8th Annual Conference on Robot Learning*, 2024a.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric, 2018.

Tianyuan Zhang, Hong-Xing Yu, Rundi Wu, Brandon Y. Feng, Changxi Zheng, Noah Snavely, Jiajun Wu, and William T. Freeman. PhysDreamer: Physics-Based Interaction with 3D Objects via Video Generation, April 2024b. URL http://arxiv.org/abs/2404.13026. arXiv:2404.13026 [cs].