[2026 Winter/Spring] KAIST URP Program

Individual Research Project Final Report

[Creative Project]

물리적으로 타당한 시뮬레이션을 위한 그라운디드 다중 객체 3D 재구성

Grounded Multi-object 3D Reconstruction for Physically Plausible Simulation

Dept: School of Computing / Name: Kyaw Ye Thu / Student No. 20220929

For Submission

Research Subject (Korean): 물리적으로 타당한 시뮬레이션을 위한 그라운디드 다중 객체 3D 재구성

Research Subject (English): Grounded Multi-object 3D Reconstruction for Physically Plausible Simulation

Research Period : January 2, 2026 ∼ June 19, 2026

Advisory Professor : 윤성의 (Signature)

Teaching Assistant : 이주민 (Signature)

As a participant(s) of the KAIST URP program, I (We) have completed the above research and hereby submit the final report on the research.

June 26, 2026

Researcher: Kyaw Ye Thu (Signature)

Contents

Abstract

Realistic physics simulation of reconstructed 3D objects has traditionally required manually assigned material properties. Recent automated methods split into scene-level reconstruction (many objects, limited dynamics) and object-level reconstruction (single object, realistic dynamics). We bridge the two by estimating per-part material properties across multi-object scenes while striving for realistic simulation. Our pipeline is training-free and needs no per-scene optimization. A vision-language model identifies each part's material class and constitutive model from multi-view images, while the continuous parameters (Young's modulus, Poisson's ratio, density, yield stress) are grounded in a curated library of measured materials, with an LLM sampling context-aware values within physically vetted ranges to avoid numerical hallucination. For simulation, we build a multi-grid Material Point Method solver that maintains a separate velocity field per object and recovers the genuine inter-object collision, rebound, and separation that single-grid MPM cannot represent. We further contribute a procedure for assembling multi-object scenes that are stable under equilibrium. From a corpus of 42 generated scenes, we report on seven representative cases spanning distinct elastic, elastoplastic, and granular behaviors and show that our estimated grounded parameters alone can produce physically plausible simulation.

Keywords: 3D Scene Reconstruction, Vision-language Model, Physics Simulation, Material Point Method

1 Introduction

With the recent success in 3D reconstruction of the world that is consistent in appearance and geometry, there has been a growing body of research in 3D reconstruction of objects that also behave realistically to the laws of physics. Traditionally, 3D objects are manually assigned values for their material properties to enable realistic physics simulation in applications such as robotics, digital twins and video games. This research domain of achieving physically consistent behaviors in 3D reconstruction do not require explicit setting of material property values, and they can be broadly categorized into two: (1) scene-level reconstruction, which focuses on the geometry of objects in the scene, and (2) object-level reconstruction, which focuses on material parameters of the object.

Work such as PhyRecon [1], CAST [2] and 3D-RE-GEN [3] strive to achieve static physical plausibility in the scene of objects they reconstruct under equilibrium forces such as gravity, contact and friction by optimizing the geometry and placement of objects. But, they do not embody any notion of weight, elasticity and plasticity of objects. On the other hand, work such as Pixie [4], PhysDreamer [5], OmniPhysGS [6] and NeRF2Physics [7] aim to enable a realistic interactivity of a single object under custom forces by optimizing its material properties such as Young' s modulus, Poisson' s ratio and density. In this regard, we can observe that while scene-level reconstruction considers many objects in a scene, its applicability in realistic physics simulation is limited, and the reverse is true for object-level reconstruction. Our work aims to bridge the gap between these two ends of a spectrum by enabling 3D reconstruction of multi-object scenes while attending to material properties of each particle in each object therein for physically plausible simulation.

Our approach is inspired by the ease with which we, as humans, infer how an object will respond to applied forces from its appearance alone. Seeing an armchair, we immediately expect the cushion to be soft and the frame to be stiff, without knowing which exact material they are made of. And when we can identify the material more precisely (whether the cushion is packed with steel coils or with foam), our estimate of its stiffness and elasticity sharpens accordingly. This capability to make a good inference stems from our prior knowledge accumulated through our lifetime of interaction with the world. Building on this analogy, our approach uses a vision-language model to infer object material types given multi-view images of the multi-object scene, treating its pretrained weights as a proxy for this accumulated prior knowledge. For continuous physical properties such as Young' s modulus, Poisson' s ratio and density, we recover them, as a range of values, through lookup in a curated material library instead of directly asking a VLM to predict.

In object-level, physics-aware 3D reconstruction, a plausible simulation can arise from two complementary factors: physical parameters assigned to each object and the configuration of the simulator itself. If the simulator can be tuned freely per scene with a bespoke configuration file, there is little need for estimated parameters to be optimal. After all, some sub-optimal combination of physical parameters and hand-tuning configuration parameters can produce the desired behavior. On the other hand, if the estimated parameters reflect the real-world

parameters well, the simulator can have a general configuration with minimal per-scene fine-tuning. Physically reasonable motion should follow from good physical parameters, not from scene-specific tuning. Prior work varies in how heavily it leans on the simulator to obtain plausible results. For example, Pixie additionally specifies per-object-class numerical settings such as a hardening coefficient, a yield stress, and a particle-filling scheme [4]. OmniPhysGS selectively applies a force on certain parts of the object to achieve the desire motion instead of applying force uniformly on the entire object [6]. Also, current open-sourced MPM (Material Point Method) simulators such as those provided by Pixie and OmniPhysGS do not natively support genuine multi-object interaction [1]. To this end, we introduce a simulator that easily generalizes across objects of differing material types and across scenes of differing object counts with minimal per-scene tuning.

The contribution of our work is threefold, listed below in order of significance.

1. We propose a training-free **method for material parameter prediction** that uses a vision-language model to identify the discrete material type while grounding each object's continuous physical properties (Young's modulus, Poisson's ratio, and density in a curated database of measured real-world materials.
2. We develop a **multi-grid Material Point Method physics simulator** that generalizes across objects of differing material types and scenes of differing object counts.
3. We introduce a **data-preparation procedure for assembling multi-object scenes** that are stable under equilibrium. These scenes are used as starting inputs in our material parameter prediction pipeline and can serve for any research requiring multi-object scene data.

2 Related Work

Physics-integrated Gaussian Simulation. Our work builds on the line of methods that simulate physics dynamics directly on a 3D Gaussian representation. PhysGaussian [8] established this paradigm by coupling a Material Point Method (MPM) solver to 3D Gaussian Splatting. While this unifies simulation and rendering, the physical parameters that govern behavior, the material type, Young's modulus, Poisson's ratio, and density, must be assigned manually by the user. This manual specification is the central bottleneck the subsequent literature, and our own work, seeks to remove. Two broad paradigms have emerged to predict these parameters automatically, feed-forward prediction and test-time optimization.

Feed-forward Material Parameter Estimation. The first paradigm predicts physical parameters in a single pass, without any per-scene optimization. NeRF2Physics [7] distills CLIP features into a language-embedded feature field and prompts a large language model to enumerate candidate materials and their property values for an object. Pixie [4] instead trains a feed-forward network that maps distilled vision–language features into dense per-point material fields consisting of discrete material-model labels and continuous material parameters. The main limitations of these two approaches are that Pixie's material vocabulary is fixed

[1] While OmniPhysGS demonstrates multi-object interaction on their webpage, this interaction is handled implicitly by the shared background grid of a single-grid MPM solver, which enforces a no-slip contact condition and provides no control over inter-object friction or separation.

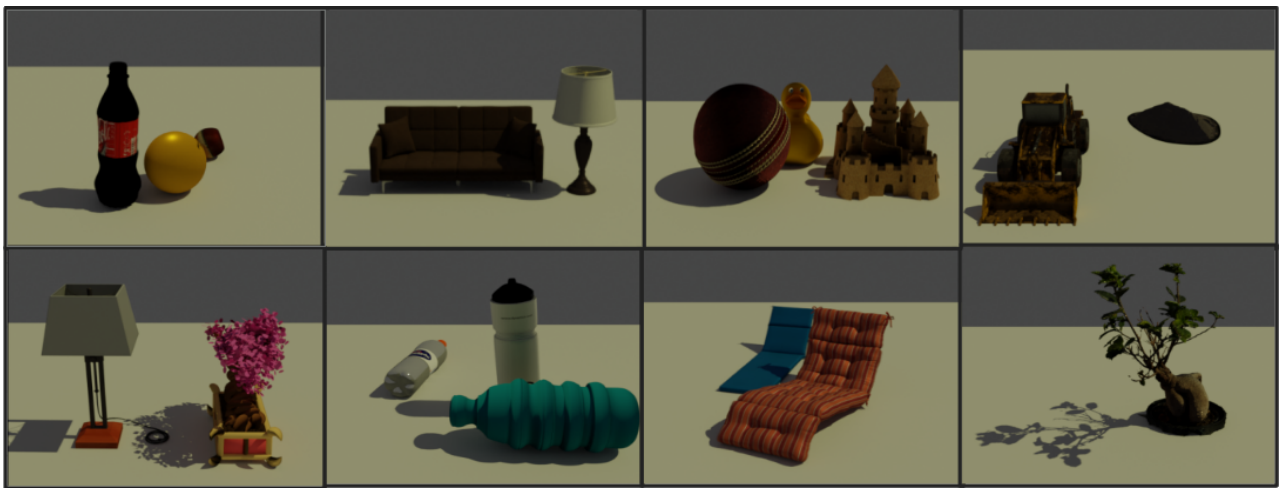

Fig. 1 Multi-object scenes with objects at a settled state under equilibrium forces. For each scene, its multi-view images are used to train a point cloud which serves the input for the material parameter estimation pipeline

by its training corpus and is shaped partly by simulation convenience rather than continuum mechanics while NeRF2Physics draws its numerical values directly from the language model, leaving them vulnerable to inconsistency and hallucination. Our pipeline is training-free and is not bounded by a fixed learned vocabulary unlike Pixie. Also, unlike NeRF2Physics, its numerical values originate from a curated, externally sourced material library rather than directly from the model itself. The model only selects a material and samples within physically vetted ranges.

Test-time optimization with generative priors. The second paradigm treats physical parameters as optimization variables and refines them per scene by distilling priors from a pretrained video-diffusion model, so that the rendered simulation matches the model's notion of plausible motion. PhysDreamer [5] and the concurrent DreamPhysics [9] introduced this approach for static objects. The former optimizes material parameters to match a reference video synthesized by an image-to-video diffusion model, and the latter distills motion priors directly through score distillation. Physics3D [6] extended the underlying material model to viscoelastic behavior and OmniPhysGS [6] generalized it to a learnable mixture over expert constitutive sub-models. As is the case with test-time optimization methods, every scene requires an optimization loop, which takes substantial time, and the recovered material identity is an implicit continuous quantity rather than an interpretable physical assignment. Since our approach is zero-shot estimation of material parameters by the combination of vision-language model and grounded lookup through a database, no per-scene optimization is required and takes much less time.

3 Method

3.1 Multi-Object Scene Preparation

We design a procedure to construct multi-object scenes automatically by sampling individual assets from two open-sourced 3D datasets and compositing them into physically settled

arrangements. First of all, given a list of object categories, individual assets are drawn from Objaverse [10] and Amazon Berkeley Objects (ABO) dataset [11]. While ABO provides clean, watertight product meshes and is used as-is, Objaverse is substantially larger but considerably noisier. So, Objaverse assets are passed through a cleaning step (filtering out low-resolution assets) before entering the asset pool. Both sources are merged into a single class-indexed asset pool, allowing a scene to mix assets drawn from either dataset.

The two datasets store meshes at arbitrary, unit-less scales. In order for the sizes of the objects in the scene to roughly reflect the real world, we query an LLM for the relative scale of each object class given the list of all object classes (Appendix C). However, real-world extents span several orders of magnitude (a soda can beside a bed), and rendering objects at true relative scale would leave small objects next to large ones too little to observe their behaviors in simulation. Given the per-object extents s_i in a scene, we compute compressed sizes s_i^γ with $\gamma = 0.35$ and map them to absolute target extents so that the largest object fills a fixed fraction f of the scene's working volume:

$$\text{scale}_i = \frac{s_i^\gamma}{\max_j s_j^\gamma} \cdot fL$$

where f is a fraction of the scene extent, L. Intuitively, the lower the γ, the more uniform the object's sizes are whereas the higher the γ, the more realistic they are.

Each scene comprises 1∼4 objects, and we ensure that the frequencies of object classes are approximately balanced across the corpus. The selected objects are composited using BlenderProc [12], with which we run a short rigid-body simulation that lets each object fall and settle into a stable resting pose after sampling collision-free initial object positions and placing the camera. Fig. 1 shows several such settled scenes exported as a GLB asset.

3.2 Material Parameters Estimation

We use a point cloud as the 3D representation of a multi-object scene, where each point is assigned material parameter values estimated by our method. While n multi-view images are used as inputs to perform grounded material parameter estimation (Section 3.2.2), RGB appearance is not sufficient for the optimal segmentation of objects and, therefore, for the assignment of parameter values to each point. As such, we additionally associate each point with both CLIP [13] and DINOv2 [14] feature descriptor and use a feature-rich point cloud as an input to perform segmentation and parameter value assignment (Section 3.2.3). The overall pipeline is shown in Fig. 2.

3.2.1 Reconstruction as Point Cloud from Multi-view Images

From posed multi-view RGB images of a multi-object scene, we reconstruct a feature-augmented radiance field following F3RM [15]. Alongside the usual color and volume density, the field predicts a view-independent feature vector at every spatial location, supervised by

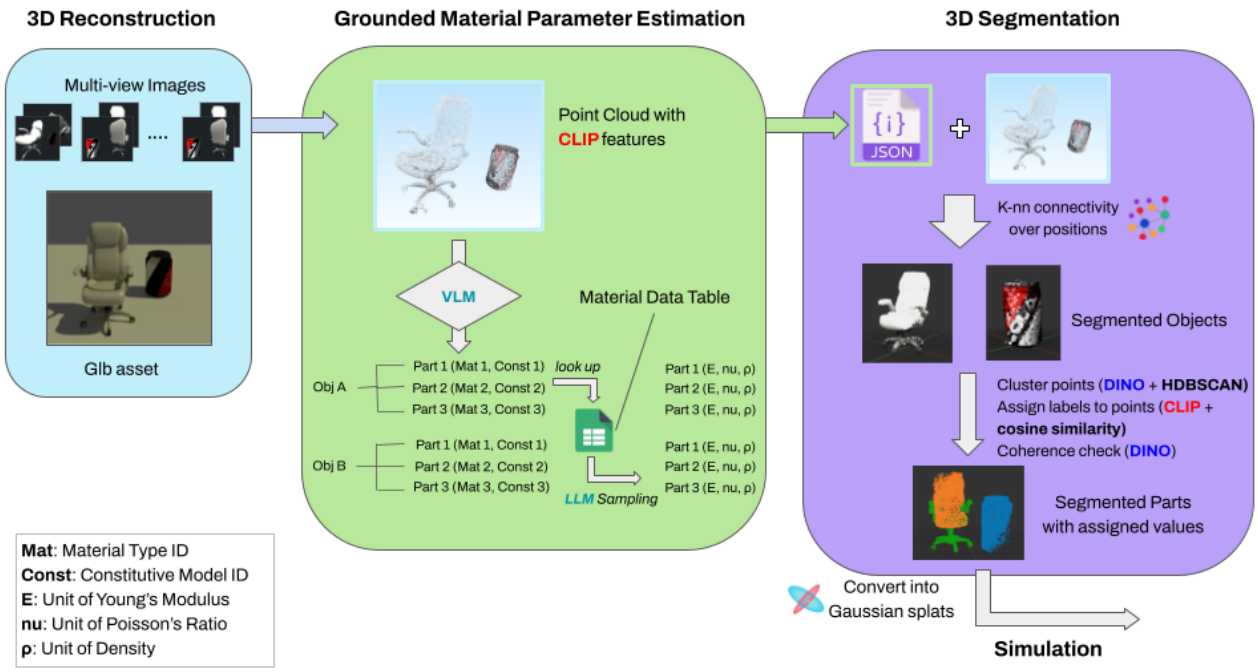

Fig. 2 Overview of the proposed material-parameter estimation pipeline. Given a GLB scene and posed multi-view RGB images, we reconstruct a feature-augmented point cloud with CLIP and DINOv2 descriptors. A VLM identifies object parts and material classes, database-grounded lookup provides physically valid parameter ranges, and segmentation assigns the resulting material parameters to each point before transferring them to Gaussian particles for simulation.

per-pixel CLIP embeddings from the training frames:

$$F_\theta : (x, d) \to (f(x), c(x, d), \sigma(x))$$

where $c \in \mathbb{R}^3$ and $\sigma \in \mathbb{R}_{\geq 0}$ are standard color and opacity outputs of NeRF [16] and $f \in \mathbb{R}^d$ is a d dimensional CLIP feature descriptor ($d = 768$).

We additionally associate each reconstructed point with a DINOv2 descriptor, but unlike distilling through volume rendering as for CLIP features, we back-project features from 2D, where each point is projected into every view in which it is visible. Then, the dense DINOv2 feature map of that view is sampled at the projected pixel, and the per-view samples are averaged into a single descriptor robust to per-view noise.

3.2.2 VLM-assisted Grounded Material Parameter Estimation

The goal is to estimate four continuous parameters (Young's modulus E, Poisson's ratio ν, density ρ, yield stress σ_y), and one discrete parameter (constitutive model ID) for each point in the point cloud. Constitutive models are physical models that determine the dynamics of different materials by providing the relationship between stress and strain, which is elaborated in Appendix B.2. Inspired by the idea of learning a constitutive model for each Gaussian particle in OmniPhysGS, we consider it as a material parameter to predict. Writing $\theta = (E, \nu, \rho, \sigma_y)$ for the parameter vector, m for the material class, I for the multi-view observations, and s for

the object- and part-level name in a string, we can formulate our objective as followings:

$$\theta \sim p(\theta \mid m^\star, s), \qquad m^\star = \arg\max_m p(m \mid I), \tag{1}$$

where we choose the single most probable material per object part, $m^\star$ by prompting a vision-language model and condition the continuous parameters on it. The two stages are described below.

Discrete Material Classification.

We prompt a vision-language model to identify distinguished objects and their parts given a randomly sampled subset of the rendered multi-view RGB images as inputs. Here, in order to have consistent distinguishing of objects and parts aligned with our goal of realistic simulation, we prompt the model for n alternative ways of distinguishing, after which we prompt a critic model to rank among them and pick the highest rated one. With the final list of object parts, we again prompt the model to assign each part a single material label from a fixed vocabulary of 22 candidate materials (metals, woods, plastics, foam, rubber, ceramics, and so on). Each predicted class deterministically determines one of five constitutive models: fixed corotated, Neo-Hookean, St. Venant–Kirchhoff (StVK), and StVK coupled with either Drucker–Prager or von Mises plasticity [2].

Database-grounded Parameter Selection.

The second conditional probability $p(\theta \mid m^\star, s)$ entails the combination of grounded lookup and LLM sampling. Here, we do not ask a model to directly regress the continuous quantities as in [17] because language models are reported to exhibit scaling and magnitude errors in zero-shot numerical regression without in-context exemplars and across quantities spanning many orders of magnitude [18] [3]. For each object part, we retrive the admissible range $[\theta_{\text{low}}, \theta_{\text{high}}]$ for the four continuous parameters using its material type, $m^\star$ as index. Then, a text-only call to a langauge model selects context-aware values *within* those ranges, conditioned on its object name and material type. This lets the same nominal material occupy different points of its range according to likely use and treatment. For example, aluminium in a drinks can is placed at low stiffness and low yield, whereas aluminium in a structural frame is placed mid-range. While we intend the sampled parameters to reflect real-world measurements for our method to be intuitive and physically faithful, Young's modulus and yield strain are adjusted for numerical stability as detailed in Appendix A.

We use `gpt-5.4` for object/part segmentation and parameter sampling, and `gpt-5-mini` as the critic model. All prompts are provided in Appendix C.

[2] Unlike in OmniPhysGS, fluids are excluded. This is because while all the five constitutive models share an elastic response parameterised by E and ν, a fluid has no shear modulus as its deviatoric stress responds to the strain rate through a viscosity, a quantity that is out of our scope

[3] Even though language models remain capable regressors once given in-context input–output pairs [17], our appraoch intentionally avoid the need to provide in-context example values so that it is generalizable for any object category that is covered by the selected set of material types and constitutive models.

3.2.3 Segmentation and Value Assignment

Given the feature-rich point cloud, we assign each point material parameter values in two hierarchical stages: a geometric scene-to-object stage that isolates individual object instances, followed by a semantic object-to-part stage that partitions each instance into its constituent parts.

Assuming that distinct objects in the scene occupy spatially disconnected regions of the point cloud, we build a symmetric k-nearest-neighbor connectivity graph over point positions ($k = 20$). Then, we extract its connected components and take each component as one object instance. Every instance inherits the CLIP and DINOv2 descriptors of its points and is matched to one of the objects identified by the VLM in Section 3.2.2 through a global (Hungarian) assignment over CLIP similarity. Each instance is then segmented into parts independently.

Within an instance, the two feature types play distinct roles: DINOv2 determines *where* part boundaries lie, while CLIP determines *what* each part is. Let $\hat{g}_i \in \mathbb{R}^{768}$ be the normalized DINOv2 descriptor at point i and $x_i \in \mathbb{R}^3$ its position. The clustering input for point i is a concatenated vector of features and coordinates, $z_i = [g;\ w\, x_i]$ where w weights position relative to appearance. HDBSCAN [19] groups $\{z_i\}$ under the Euclidean metric, whose squared distance splits into an appearance and a geometry term,

$$\|z_i - z_j\|_2^2 = \underbrace{\left\|\hat{g}_i - \hat{g}_j)\right\|_2^2}_{\text{appearance (DINOv2)}} + w^2 \underbrace{\|\bar{x}_i - \bar{x}_j\|_2^2}_{\text{geometry}},$$

The resulting k clusters are labeled by CLIP in two passes. In the first, each cluster's CLIP prototype is formed as a DINOv2-weighted average of its per-point CLIP features [4] and matched to the part-name queries by cosine similarity. During the second pass, we ensure that clusters with the same label are similar in DINOv2 feature space by relabeling clusters. Finally, with each point being assigned an object part, it receives the parameter vector $\theta = (E, \nu, \rho, \sigma_y)$ and constitutive-model ID already attached to its part.

3.3 Physics Simulation

To simulate physics, we use the Material Point Method (MPM), a hybrid Lagrangian–Eulerian scheme that carries state on particles while resolving momentum on a background grid. Although the MPM solver can take a point cloud output by NeRF, we use a Gaussian Splatting model with each Gaussian splat as a MPM particle for the ease of use. Hence, we separately learn a Gaussian splatting model from posed multi-view RGB images and transfer the material properties from NeRF point cloud into the Gaussian splatting model with nearest neighbor interpolation. With Gaussian splats of inherited material properties and external force specification as inputs, we simulate the transformation and deformation of each particle.

Standard MPM resolves momentum on a single, shared background grid with a single-valued velocity field at each node. This automatically precludes interpenetration, but it does so by imposing an inherent no-slip, no-separation contact. i.e, objects that share a node are

[4] For a weighted average of the CLIP vectors of the points in the cluster, we use a DINOv2-weighted average instead of a plain average. Intuitvely, each point's weight is how closely its DINO feature matches the cluster's average DINO feature. Points that look typical of the cluster gets more contribution, while points that look atypical such as boundary noise get downweighted.

effectively fused, unable to slide past one another or to separate once in contact. For multi-body dynamics, it means that distinct objects would move together rather than collide, rebound, and part. To recover true inter-object independence, we maintain a separate grid per object and resolve contact explicitly at shared nodes, following a multi-velocity-field contact algorithm introduced in [20]. Essentially, we implement the multi-grid MPM simulator on top of the standard MPM simulator from OmniPhysGS.

4 Experiments

Dataset. The dataset is produced from the data preparation pipeline described in Section 3.1 and consists of 42 multi-object scenes spanning 15 unique object classes. For the qualitative results reported here we use a subset of seven scenes, leaving the evaluation of the remaining scenes for the extended experiments as future work. These seven scenes are selected to cover all the five constitutive models in our taxonomy (Appendix B.2), object counts of one to three per scene, and several interaction types (inter-object collision, free fall onto the ground, free fall onto another object, and chain-reaction collisions).

Simulation Details. Each simulation runs for 80–180 frames on a single NVIDIA RTX 5070 GPU. Gravity acts on every object in the scene, while wind is applied per-object —to all, some, or none— according to the interaction we wish to elicit. Each scene takes 30–150 seconds for material-parameter estimation, depending on the number of objects it contains, and two to ten minutes to simulate, depending on the timestep size dictated by the CFL condition (Appendix A).

4.1 Constitutive Behaviors

Physics simulations of the 7 representative scenes are shown in Fig. 3, where the behavior of each constitutive model assigned by our pipeline becomes visible. The per-part constitutive model is visualized by color in Fig. 5, alongside material type, Young's modulus, Poisson's ratio, and density.

- In the *duck-on-mud scene*, the mud yields under the duck's weight while the duck rests on the deformed surface undeformed, which is characteristic of a granular constitutive law paired with an elastic one.
- In the *duck-and-can scene*, the rubber duck deforms on contact and recovers its shape while the can yields and retains a permanent dent. It shows the contrast between elastic and elastoplastic constitutive laws.
- In the *vase-ball-armchair* scene, force propagates through a chain of distinct objects (a vase into a ball into an armchair) rather than fusing them into a single body.
- In the *lamp-and-couch scene*, the lamp preserves its shape on impact while the couch cushion undergoes large compressive deformation and rebounds as the lamp lifts off. It is characteristic of a soft elastic material with no plastic threshold.

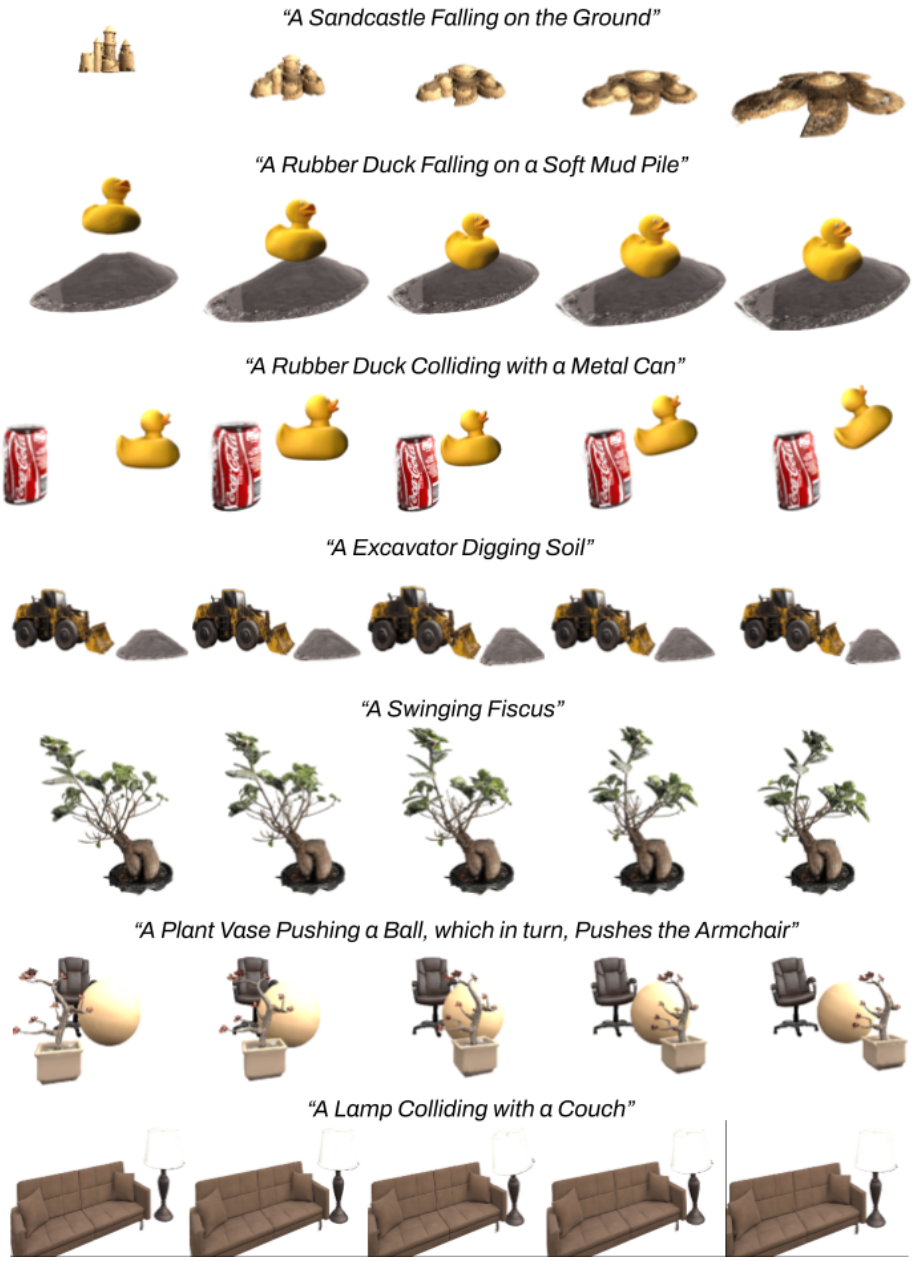

Fig. 3 Qualitative simulation results of seven representative multi-object scenes. Each row shows a temporal sequence under gravity and/or externally applied forces, illustrating different material behaviors such as elastic recovery, plastic deformation, granular yielding, and inter-object collision/rebound.

4.2 Qualitative Comparison with Baseline Methods

We compare against PhysDreamer and OmniPhysGS using their published visualizations rather than re-implementing them as shown in Fig 4. In the published examples, PhysDreamer appears to produce limited multi-object interaction, where the metal can does not move and the soil/mud pile does not visibly deform. On the other hand, both OmniPhysGS and our method produce multi-object dynamics, albeit differently in some scenes.

- For the *swinging-ficus* scene, OmniPhysGS keeps the pot stationary through an explicit boundary condition: a cuboid region over the base is pinned to zero grid velocity, while the driving impulse is applied uniformly to the entire object. Our method requires no such

Fig. 4 Qualitative comparison of 3D dynamics with PhysDreamer and OmniPhysGS

constraint, and yet only the foliage sways while the pot stays effectively static when the whoel object is applied uniform force. We attribute this to grounding as the pot's high Young's modulus and density resist the very perturbation that makes the softer foliage yield.

- For the *duck-and-can* scene, both methods produce a dent, but at different locations. In the OmniPhysGS example, the dent appears displaced from the point of impact, whereas our method localizes it at the contact point. We attribute this localization to our multi-velocity-field contact formulation, which transfers momentum only through overlapping per-object particle neighborhoods, rather than through a single shared grid, where solid–solid contact tends toward an implicit no-slip condition.

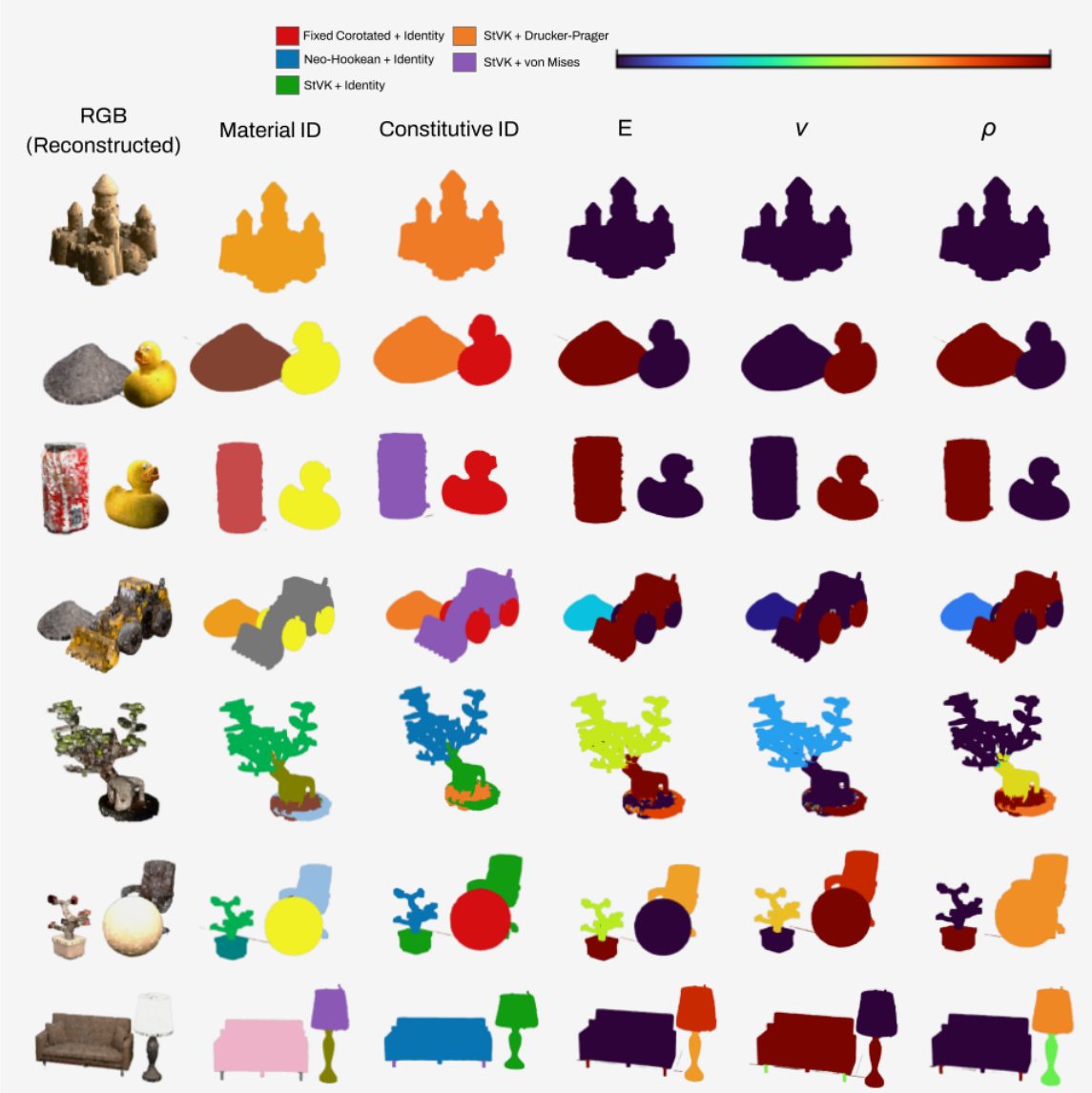

Fig. 5 Visualization of estimated material type, constitutive model, Young's modulus (E), Poisson's ratio (ν), and density (ρ) of scenes in Fig 3. For each scene, *continuous values*, E, ν, and ρ, are independently min–max normalized and mapped to the turbo colormap (blue to red denotes low to high). Their colors are therefore relative within each row and not comparable across scenes or across properties. However, *discrete properties*, material ID and constitutive ID, are comparable across scenes (The same color across rows represent the same ID).

5 Future Research Plan

5.1 Evaluation at Corpus Scale

Our immediate next step is to extend the evaluation from the 7 representative scenes to all 42 produced by the preparation pipeline of Section 3.1. Because the corpus is class-balanced by construction, we will report results broken down per material class and per constitutive model rather than as the aggregate, illustrative cases shown so far. A second direction of scaling is the object vocabulary itself: the pipeline currently spans fifteen object classes, and because it is training-free, it is easily extendable to new classes, provided their behaviors are captured by our constitutive models. We therefore plan to enlarge the dataset with further object classes too.

5.2 Quantitative Evaluation

Physical plausibility admits no single ground-truth metric when a scene has no reference dynamics, so we plan to evaluate it in two ways: an objective benchmark that isolates the parameter-estimation stage and a holistic judgment of the simulated dynamics as a whole. For the objective evaluation, we plan to evaluate our estimated density on the ABO-500 mass benchmark curated by NeRF2Physics. The benchmark is a natural fit, since ABO is already one of the two data sources of our dataset and its objects are therefore in-distribution for our pipeline. Since ABO-500 releases ground-truth object masses, we integrate our estimated density over the reconstructed volume to obtain a predicted mass and compare it against the released value. To assess the dynamics holistically, we plan to use a multimodal model as an automatic judge of physical plausibility of our simulated scenes, following recent video-generation physics benchmarks [21, 22].

5.3 Re-implemented Baseline Comparison

Currently, we use published visualization of only PhysDreamer and OmniPhysGS as qualitative comparison against baseline methods. We plan to actually set up more baseline methods and obtain material property values for more scenes, after which we simulate them with our own multi-grid MPM simulator. The rendered dynamics will be compared visually and scored by the multimodal-judge protocol described above. OmniPhysGS operates on multi-object scenes directly (via its shared background grid) and can be expected to run unmodified. Pixie, NeRF2Physics, and PhysDreamer are object-level methods, so we run each pipeline on every object in a scene separately and stitch the per-object results back into the shared scene before simulating. This lets all methods be evaluated on the same multi-object inputs.

6 Conclusion

We present a training-free pipeline that closes the gap between scene-level reconstruction, which captures many objects but only static geometry, and object-level reconstruction, which captures real dynamics but only for a single object. We separate the problem of estimating mateiral properties of multi-object scene into a discrete and a continuous stage: a vision-language model identifies each part's material class and, through it, a constitutive model, while the continuous parameters are drawn from a curated library of measured materials rather than emitted by the model directly, which removes the numerical hallucination that besets language-model parameter prediction. As our secondary contribution, we also developed a multi-grid Material Point Method solver which can recover the inter-object collision, rebound, and separation that a single shared grid cannot represent. Lastly, we contributed a data-preparation procedure that assembles multi-object scenes settled under equilibrium, which serve as inputs to our pipeline and can support any work requiring multi-object scene data.

Together, these components produce physically distinguishable behavior from appearance alone and without per-scene tuning of the simulator. For example, stiff objects resist forces that deform soft ones, elastic parts recover while plastic parts retain their deformation, and distinct objects collide and separate rather than fusing. Nonetheless, the range of behaviors we capture

is bounded by our constitutive taxonomy, whose five models all share an elastic response parameterized by E and ν. Materials whose stress depends on strain rate rather than strain, such as fluids and viscoelastic media, therefore fall outside it. A natural direction is to add constitutive branches for viscous and viscoelastic behavior, along with the physical parameters they require, such as viscosity, and to enrich the present 22-material vocabulary with finer-grained granular and other classes. Because both the recognition and the grounding stages are training-free, this broadening amounts to extending the material library and its constitutive mapping rather than retraining, which we see as the most direct route to generalizing the framework.

References

[1] Ni, J., Chen, Y., Jing, B., Jiang, N., Wang, B., Dai, B., Li, P., Zhu, Y., Zhu, S.-C., Huang, S.: Phyrecon: Physically plausible neural scene reconstruction. (2024)

[2] Yao, K., Zhang, L., Yan, X., Zeng, Y., Zhang, Q., Yang, W., Xu, L., Gu, J., Yu, J.: CAST: Component-Aligned 3D Scene Reconstruction from an RGB Image (2025). https://arxiv.org/abs/2502.12894

[3] Sautter, T., Dihlmann, J.-N., Lensch, H.P.A.: 3D-RE-GEN: 3D Reconstruction of Indoor Scenes with a Generative Framework (2025). https://arxiv.org/abs/2512.17459

[4] Le, L., Lucas, R., Wang, C., Chen, C., Jayaraman, D., Eaton, E., Liu, L.: Pixie: Fast and generalizable supervised learning of 3d physics from pixels. arXiv preprint arXiv:2508.17437 (2025)

[5] Zhang, T., Yu, H.-X., Wu, R., Feng, B.Y., Zheng, C., Snavely, N., Wu, J., Freeman, W.T.: PhysDreamer: Physics-based interaction with 3d objects via video generation. In: European Conference on Computer Vision (2024). Springer

[6] Lin, Y., Lin, C., Xu, J., Mu, Y.: OmniPhysGS: 3D Constitutive Gaussians for General Physics-Based Dynamics Generation (2025). https://arxiv.org/abs/2501.18982

[7] Zhai, A.J., Shen, Y., Chen, E.Y., Wang, G.X., Wang, X., Wang, S., Guan, K., Wang, S.: Physical Property Understanding from Language-Embedded Feature Fields (2024). https://arxiv.org/abs/2404.04242

[8] Xie, T., Zong, Z., Qiu, Y., Li, X., Feng, Y., Yang, Y., Jiang, C.: PhysGaussian: Physics-Integrated 3D Gaussians for Generative Dynamics (2024). https://arxiv.org/abs/2311.12198

[9] Huang, T., Zhang, H., Zeng, Y., Zhang, Z., Li, H., Zuo, W., Lau, R.W.H.: DreamPhysics: Learning Physics-Based 3D Dynamics with Video Diffusion Priors (2024). https://arxiv.org/abs/2406.01476

[10] Deitke, M., Liu, R., Wallingford, M., Ngo, H., Michel, O., Kusupati, A., Fan, A., Laforte, C., Voleti, V., Gadre, S.Y., VanderBilt, E., Kembhavi, A., Vondrick, C., Gkioxari, G., Ehsani, K., Schmidt, L., Farhadi, A.: Objaverse-XL: A Universe of 10M+ 3D Objects (2023). https://arxiv.org/abs/2307.05663

[11] Collins, J., Goel, S., Deng, K., Luthra, A., Xu, L., Gundogdu, E., Zhang, X., Vicente, T.F.Y., Dideriksen, T., Arora, H., Guillaumin, M., Malik, J.: ABO: Dataset and Benchmarks for Real-World 3D Object Understanding (2022). https://arxiv.org/abs/2110.06199

[12] Denninger, M., Sundermeyer, M., Winkelbauer, D., Zidan, Y., Olefir, D., Elbadrawy, M., Lodhi, A., Katam, H.: BlenderProc (2019). https://arxiv.org/abs/1911.01911

[13] Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., Sutskever, I.: Learning Transferable Visual Models From Natural Language Supervision (2021). https://arxiv.org/abs/2103.00020

[14] Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., Assran, M., Ballas, N., Galuba, W., Howes, R., Huang, P.-Y., Li, S.-W., Misra, I., Rabbat, M., Sharma, V., Synnaeve, G., Xu, H., Jegou, H., Mairal, J., Labatut, P., Joulin, A., Bojanowski, P.: DINOv2: Learning Robust Visual Features without Supervision (2024). https://arxiv.org/abs/2304.07193

[15] Shen, W., Yang, G., Yu, A., Wong, J., Kaelbling, L.P., Isola, P.: Distilled Feature Fields Enable Few-Shot Language-Guided Manipulation (2023). https://arxiv.org/abs/2308.07931

[16] Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis (2020). https://arxiv.org/abs/2003.08934

[17] Vacareanu, R., Negru, V.-A., Suciu, V., Surdeanu, M.: From Words to Numbers: Your Large Language Model Is Secretly A Capable Regressor When Given In-Context Examples (2024). https://arxiv.org/abs/2404.07544

[18] Pavlidis, N., Perifanis, V., Symeonidis, S., Efraimidis, P.S.: Large Language Models as Universal Predictors? An Empirical Study on Small Tabular Datasets (2025). https://arxiv.org/abs/2508.17391

[19] Malzer, C., Baum, M.: A hybrid approach to hierarchical density-based cluster selection. In: 2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), pp. 223–228. IEEE, ??? (2020). https://doi.org/10.1109/mfi49285.2020.9235263 . http://dx.doi.org/10.1109/MFI49285.2020.9235263

[20] Huang, P., Zhang, X., Ma, S., Huang, X.: Contact algorithms for the material point method in impact and penetration simulation. International Journal for Numerical Methods in Engineering **85**(4), 498–517 (2011) https://doi.org/10.1002/nme.2981 https://onlinelibrary.wiley.com/doi/pdf/10.1002/nme.2981

[21] Bansal, H., Lin, Z., Xie, T., Zong, Z., Yarom, M., Bitton, Y., Jiang, C., Sun, Y., Chang, K.-W., Grover, A.: VideoPhy: Evaluating Physical Commonsense for Video Generation (2024). https://arxiv.org/abs/2406.03520

[22] Meng, F., Liao, J., Tan, X., Shao, W., Lu, Q., Zhang, K., Cheng, Y., Li, D., Qiao, Y., Luo, P.: Towards World Simulator: Crafting Physical Commonsense-Based Benchmark for Video Generation (2024). https://arxiv.org/abs/2410.05363

[23] MakeItFrom: About. https://www.makeitfrom.com/about Accessed 2026-06-24

[24] Stomakhin, A., Howes, R., Schroeder, C., Teran, J.M.: Energetically consistent invertible elasticity. In: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation. SCA '12, pp. 25–32. Eurographics Association, Goslar, DEU (2012)

[25] Bonet, J., Wood, R.D.: Nonlinear Continuum Mechanics for Finite Element Analysis. Cambridge University Press, Cambridge, UK (1997)

[26] Klár, G., Gast, T., Pradhana, A., Fu, C., Schroeder, C., Jiang, C., Teran, J.: Drucker-prager elastoplasticity for sand animation. ACM Trans. Graph. **35**(4) (2016) https://doi.org/10.1145/2897824.2925906

[27] Drucker, D.C., Prager, W.: Soil mechanics and plastic analysis or limit design. Quarterly of Applied

Mathematics **10**(2), 157–165 (1952) https://doi.org/10.1090/qam/48291

[28] Wu, C., Huang, L., Zhang, Q., Li, B., Ji, L., Yang, F., Sapiro, G., Duan, N.: GODIVA: Generating Open-DomaIn Videos from nAtural Descriptions (2021). https://arxiv.org/abs/2104.14806

[29] Yue, Y., Smith, B., Chen, P.Y., Chantharayukhonthorn, M., Kamrin, K., Grinspun, E.: Hybrid grains: adaptive coupling of discrete and continuum simulations of granular media. ACM Trans. Graph. **37**(6) (2018) https://doi.org/10.1145/3272127.3275095

[30] Barbič, J., James, D.L.: Real-time subspace integration for st. venant-kirchhoff deformable models. In: ACM SIGGRAPH 2005 Papers, pp. 982–990. ACM, ??? (2005). https://doi.org/10.1145/1073204.1073291

[31] Mises, R.: Mechanik der festen körper im plastisch-deformablen zustand. Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse, 582–592 (1913)

[32] Hu, Y., Fang, Y., Ge, Z., Qu, Z., Zhu, Y., Pradhana, A., Jiang, C.: A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. ACM Trans. Graph. **37**(4) (2018) https://doi.org/10.1145/3197517.3201293

[33] Huang, Z., Hu, Y., Du, T., Zhou, S., Su, H., Tenenbaum, J.B., Gan, C.: PlasticineLab: A Soft-Body Manipulation Benchmark with Differentiable Physics (2021). https://arxiv.org/abs/2104.03311

Appendix A CFL Condition for Numerical Stability in Simulation

Our explicit MPM solver advances stress in time assuming information propagates only between neighboring grid cells within a single step. This is stable only if no elastic wave travels farther than a fraction of one cell per timestep, known as the Courant–Friedrichs–Lewy (CFL) condition. The wave speed dictates how fast stress information propagates through the material. A simple estimate of the elastic wave speed is $c = \sqrt{E/\rho}$, but the relevant quantity in a 3D continuum is the longitudinal (P-wave) speed, which carries the full dilatational stiffness,

$$c_p \;=\; \sqrt{\frac{E\,(1-\nu)}{\rho\,(1+\nu)(1-2\nu)}}. \tag{A1}$$

The CFL condition requires this wave to cross at most a fraction $s \in (0, 1]$ of a cell per step,

$$c_p\,\Delta t \;\le\; s\,\Delta x, \tag{A2}$$

with grid spacing Δx and timestep Δt. Violating Eq. (A2) lets the wave jump across multiple cells in one step. Then, the solver misses intermediate states and numerical errors explode.

Since the sampled E in Sec 3 may violate Eq. (A2), we cap its value at the largest stiffness admitting a stable P-wave speed at the chosen ρ and ν,

$$E_{\max} \;=\; \rho\left(\frac{s\,\Delta x}{\Delta t}\right)^2 \frac{(1+\nu)(1-2\nu)}{1-\nu}, \tag{A3}$$

To leave the plastic response qualitatively unchanged under this cap, we scale the yield stress by the same factor, $\sigma_{y,\mathrm{sim}} = \sigma_y\,(E_{\mathrm{sim}}/E)$, which preserves the yield strain $\sigma_y/2\mu$ since the shear modulus μ is proportional to E at fixed ν.

Appendix B Material Library

B.1 Curated Material Library

We curate a library of material property values (Table B1) that our pipeline retrieves and from which the LLM samples. The values are aggregated from standard engineering and materials references, most frequently MakeItFrom [23].

B.2 Constitutive Models

Each segmented part is simulated with a standard MPM material model, selected by a discrete constitutive identifier $c \in \{0, \ldots, 4\}$ that pairs an elastic energy density with a plastic return mapping. We adopt established formulations [24–33] just as in OmniPhysGS.

Given Young's modulus E and Poisson's ratio ν, the Lamé parameters fed to the simulator are

$$\mu = \frac{E}{2(1+\nu)}, \qquad \lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}. \tag{B4}$$

Material	Mat. ID	Const. ID	E (GPa)	ν	ρ (g/cm^3)	σ_y (MPa)
gold	1	4	77.2	0.42–0.44	19.29	120
silver	2	4	71–82	0.37–0.39	10.5	54–730
magnesium	3	4	45–47	0.35–0.40	1.74	80–130
copper	4	4	110	0.34–0.36	8.94	80–150
aluminium	5	4	68	0.33–0.35	2.7	40–210
plasticine	6	4	0.001–0.01	0.40–0.499	1.6–1.9	0.05–0.2
steel	7	4	190–200	0.30–0.31	7.7–8.05	230–510
soil_clay	8	3	0.002–0.1	0.3–0.5	1.8–2.6	—
cast-iron	9	2	180–190	0.28–0.31	7.5–8.0	—
sand	10	3	0.03–0.08	0.20–0.455	1.4–1.6	—
concrete	11	2	30	0.15–0.25	1.3–2.0	—
glass	12	2	70–71	0.23	2.4	—
wood	13	2	12–16	0.37–0.39	0.7	—
plastics-HDPE	14	2	1	0.45–0.50	1.0–1.3	—
plastics-LDPE	15	1	0.3–0.5	0.33–0.35	0.92	—
plastics-PP	16	2	0.9–1.1	0.35–0.40	0.9–1.2	—
polystyrene	17	2	1.9–2.9	0.40–0.41	1.0	—
foam	18	1	0.0025–0.007	0.35–0.40	0.03	—
rubber	19	0	0.01–0.1	0.4999	1.0–1.2	—
leaves-flowers	20	1	0.2–0.5	0.4	0.1–0.2	—
snow	21	3	0.0002–0.02	0.15–0.25	0.1–0.35	—
ceramic	22	2	67–80	0.21–0.23	2.4–2.7	—

Table B1 Curated material library. A single value denotes a fixed quantity. Constitutive ID: 0 = Fixed Corotated + Identity, 1 = Neo-Hookean + Identity, 2 = StVK + Identity, 3 = StVK + Drucker–Prager, 4 = StVK + von Mises (Sec **??**). "—" denotes yield strength not used by the assigned model (stored as 0).

Hyperelasticity Energy Density Functions

We express each model through the first Piola–Kirchhoff stress $\mathbf{P} = \partial\Psi/\partial\mathbf{F}$, mapping the deformation gradient $\mathbf{F}$ to $\mathbf{P}$. Let $\mathbf{F} = \mathbf{RS}$ be the polar decomposition, $\mathbf{F} = \mathbf{U\Sigma V}^\top$ the SVD, and $J = \det \mathbf{F}$.

$$\text{Fixed Corotated [24]:} \quad \mathbf{P} = 2\mu(\mathbf{F} - \mathbf{R}) + \lambda J(J-1)\mathbf{F}^{-\top}, \tag{B5}$$

$$\text{Neo-Hookean [25]:} \quad \mathbf{P} = \mu(\mathbf{F} - \mathbf{F}^{-\top}) + \lambda \log(J)\mathbf{F}^{-\top}, \tag{B6}$$

$$\text{StVK [26, 30]:} \quad \mathbf{P} = \mathbf{U}\big(2\mu\boldsymbol{\Sigma}^{-1}\ln\boldsymbol{\Sigma} + \lambda\,\mathrm{tr}(\ln\boldsymbol{\Sigma})\boldsymbol{\Sigma}^{-1}\big)\,\mathbf{V}^\top. \tag{B7}$$

Plasticity Return Functions

A return mapping $\psi(\cdot)$ corrects the trial deformation gradient (the trial subscript is omitted). With $\boldsymbol{\epsilon} = \log(\boldsymbol{\Sigma})$:

$$\text{Identity:} \quad \psi(\mathbf{F}) = \mathbf{F}, \tag{B8}$$

$$\text{Drucker–Prager [26–29]:} \quad \psi(\mathbf{F}) = \mathbf{U}\,\mathcal{Z}(\boldsymbol{\Sigma})\,\mathbf{V}^\top,\ \mathcal{Z}(\boldsymbol{\Sigma}) = \begin{cases} \mathbf{I}, & \mathrm{sum}(\boldsymbol{\epsilon}) > 0, \\ \boldsymbol{\Sigma}, & \delta\gamma \le 0 \wedge \mathrm{sum}(\boldsymbol{\epsilon}) \le 0, \\ \exp\big(\boldsymbol{\epsilon} - \delta\gamma\frac{\hat{\boldsymbol{\epsilon}}}{\|\hat{\boldsymbol{\epsilon}}\|}\big), & \text{otherwise}, \end{cases} \tag{B9}$$

$$\text{von Mises [31–33]:} \quad \psi(\mathbf{F}) = \mathbf{U}\,\mathcal{Z}(\boldsymbol{\Sigma})\,\mathbf{V}^\top,\ \mathcal{Z}(\boldsymbol{\Sigma}) = \begin{cases} \boldsymbol{\Sigma}, & \delta\gamma \le 0, \\ \exp\!\big(\boldsymbol{\epsilon} - \delta\gamma \frac{\hat{\boldsymbol{\epsilon}}}{\|\hat{\boldsymbol{\epsilon}}\|}\big), & \text{otherwise.} \end{cases} \tag{B10}$$

The following table lists the five constitutive models used in our material library. Of the nine possible elasticity × plasticity combinations, we retain the five that correspond to distinct material behaviors in our vocabulary since the rest are redundant for the materials we model.

c	Elasticity + Plasticity	Behavior
0	Fixed Corotated + Identity	soft elastic, shape-recovering
1	Neo-Hookean + Identity	stretchy elastic
2	StVK + Identity	stiff / quasi-rigid elastic
3	StVK + Drucker–Prager	granular, shear-yielding
4	StVK + von Mises	plastic, permanent deformation

Table B2 Constitutive identifiers and their material–behavior mapping

Appendix C Prompts for (Vision) Language Models

The following are three prompts for three separate call to (vision) language models.

Listing 1 System Prompt for VLM to Segment Objects and Parts in a Scene

```
You will be provided with different views of a multi-object scene and tasked with segmenting the
    multi-object scene into objects and their parts.

### Objects vs Parts
- Objects: geometrically distinct, separable entities (a duck, a can, a tree, a pile of sand).
- Parts: subdivisions of one object with different behavioral/material properties (geometrically
    continuous but behaviorally distinct).

### When to subdivide
Subdivide ONLY when parts have obvious behavioral differences under forces
   Tree → pot, trunk, leaves (different stiffness, different wind response)
   Chair with cushion → frame, cushion (rigid vs soft)
   Soda can → DON'T split into top/body/label (same behavior)
   Rubber duck → DON'T split into head/body (same material)

Usually one to three parts per object in the primary `parts` list.

### Constraints
- Only include parts visible in the images.
- Parts within a single query list must not overlap and must cover the whole object.

Specifically, your job is to:
1. Identify all distinct objects in the scene. If there is no space between two entities (e.g., a
    plant in a pot), treat them as one object.
2. For each object, identify its high-level parts that are visible in the images.
   Describe each part as descriptively as possible (color, material-appearance,
   shape, common object-part name).
3. For each object with more than one part, propose {num_alternatives} alternative decompositions
   ("alternative_queries").

## CRITICAL: alternative_queries SHOULD VARY IN NUMBER OF PARTS
```

````
The whole point of alternatives is to let a downstream CLIP critic pick the granularity that is
    most visually separable. If all your alternatives have the same part count, the critic has
    nothing to choose between.

Do NOT output alternatives that all have the same number of parts.

## Output format

```json
{
    "objects": [
        {
            "object_id": "obj_1",
            "object_name": "descriptive name",
            "parts": ["part_name_1", "part_name_2"],
            "alternative_queries": [
                ["part_name_B1", "part_name_B2", "part_name_B3"],
                ["part_name_C1", "part_name_C2"],
                ["part_name_D1", "part_name_D2", "part_name_D3", "part_name_D4"]
            ]
        }
    ]
}
```
If an object has only one visually distinguishable part, set
"parts": ["<whole object description>"] and "alternative_queries": [].

## Example

```json
{
    "objects": [
        {
            "object_id": "obj_1",
            "object_name": "potted ficus",
            "parts": ["vase filled with soil", "brown wooden trunk", "green leafy canopy"],
            "alternative_queries": [
                ["clay pot filled with soil", "plant body"],
                ["ceramic pot", "plant stem", "green leaves"],
                ["vase with dirt", "tree trunk", "bushy foliage"]
            ]
        },
        {
            "object_id": "obj_2",
            "object_name": "office chair",
            "parts": ["black metal base frame", "grey foam seat cushion"],
            "alternative_queries": [
                ["base frame", "cushion"],
                ["chair legs", "chair backrest", "seat cushion"],
                ["steel frame", "padded cushion"]
            ]
        },
        {
            "object_id": "obj_3",
            "object_name": "soccer ball",
            "parts": ["ball body"],
            "alternative_queries": []
        }
    ]
}
```
````

Listing 2 System Prompt for VLM to Identify Material Types given Object Parts

```
You will be provided with:
      - Multiple views of a scene.
      - A list of segmented objects, each with a fixed list of `parts`.

    The part decomposition is FINAL - do not alter the given parts.

    For each part, determine the material from the provided list, {material_list}.

    ## Output format

```json
    {
        "objects": [
            {
                "object_id": "obj_1",
                "object_name": "potted ficus",
                "material_dict": {
                    "clay pot filled with soil": {"material": "fired clay", "constitutive_id": 2},
                    "brown wooden trunk":        {"material": "wood",       "constitutive_id": 2},
                    "green leafy canopy":        {"material": "leaves/plant tissues", "
    constitutive_id": 0}
                }
            }
        ]
    }
```

    Use exactly the part-name strings you are given as keys of `material_dict`.
```

Listing 3 System prompt for LLM to sample values from material property ranges

```
You are a physical material property estimator. Given an object and its parts, you will pick
    context-aware values for Young's modulus (E), Poisson's ratio (nu), density (rho), and yield
    stress (sigma_y) for each unique material used in the object. Your selections MUST lie within
    the provided ranges.

Goal: be more informative than uniform random sampling by using the object's likely treatment,
    condition, and use case to bias your choice within each range.

Key reasoning patterns (the same nominal material varies widely by treatment):

- Aluminum:
    * Soda cans, foil, soft sheet      -> low yield (~30-100 MPa), low E end.
    * Cookware, structural frames      -> mid (~150-250 MPa).
    * Aerospace parts, hardened tools  -> high (~400-500 MPa).
- Steel:
    * Paperclips, mild sheet, rebar    -> low (~200-350 MPa).
    * Structural beams, pots           -> mid (~350-500 MPa).
    * Knives, springs, tools           -> high (~700-1500 MPa).
- Wood:
    * Packing crates, soft pine        -> low E and density (softwood).
    * Furniture, oak, generic hardwood -> mid.
    * Dense exotic hardwood, mallets   -> high.
- Plastic:
    * Thin film, packaging, bags       -> low E.
    * Rigid containers, household      -> mid.
    * Engineering plastic (PA, PC)     -> high.
- Foam:
    * Seat/back cushion, soft pillow   -> very low E and density.
    * Packing foam, EVA                -> low-mid.
    * Closed-cell rigid foam           -> higher.
- Rubber/elastomer: nu near 0.49-0.5 (nearly incompressible).
- Ceramic/glass: pick consistent values; brittle, high E, low ductility.

General rules:
```

````
- Density usually varies less than E or sigma_y; still bias by context (cast vs wrought, hardwood
    vs softwood, dense vs aerated foam).
- Poisson's ratio is most stable; default to mid-range unless context is decisive (rubber -> ~0.49;
     cork -> ~0.0; metals -> ~0.30).
- Be internally CONSISTENT: if you pick the high end of E for a material (suggesting hardened/
    treated condition), pick a similarly high sigma_y.
- Use the part NAMES (not just the material name) for context.
  E.g. "soda can" vs "machined bracket" both made of "aluminium".

Output JSON only, no preamble, no commentary outside the JSON:

```json
{
  "materials": {
    "<material_name>": {
      "E": <float, in Pa>,
      "nu": <float>,
      "density": <float, in kg/m^3>,
      "sigma_y": <float, in Pa>,
      "reasoning": "<one short sentence>"
    }
  }
}
```

Every material listed in the input MUST appear in your output. Every value MUST be within the
    inclusive bounds of the provided range.
````

Listing 4 System prompt for LLM to estimate real-world extent of objects

```
You are a 3D scene layout assistant. For each object class provided,
estimate the typical real-world maximum extent in metres (i.e. the
longest dimension of a representative instance).

Reply with ONLY a JSON object mapping each class name to a float (metres).
No markdown fences, no commentary. Example:
{"chairs": 0.9, "soda_cans": 0.12}
```