Adaptive Volumetric Mechanical Property Fields Invariant to Resolution

Rishit Dagli [1 2] **Donglai Xiang** [1] **Vismay Modi** [1] **Xuning Yang** [1]
Gavriel State [1] **David I.W. Levin** [1 2] **Maria Shugrina** [1]
https://research.nvidia.com/labs/sil/projects/adavomp/

Abstract

Accurate mechanical properties (or materials) Young's modulus (E), Poisson's ratio (ν) and density (ρ) are essential for reliable physics simulation of digital worlds, but most 3D assets lack this information. We propose ADAVOMP, a method for predicting accurate dense spatially-varying (E, ν, ρ) for input 3D objects across representations, improving the resolution, accuracy, and memory efficiency over the state-of-the-art. The foundation of our technique is a sparse and *adaptive* voxel structure SAV that efficiently represents both the input 3D shape and the material field output. We replace the fixed-voxel model of the most accurate prior method, VoMP, with a novel sparse transformer encoder-decoder model that learns to generate a unique SAV autoregressively for every input shape to represent its materials, achieving a resolution $16^3\times$ higher than prior art. Experiments show that ADAVOMP estimates more accurate volumetric properties, even with lesser test-time compute than all prior art. This allows us to convert high-resolution complex 3D objects into simulation-ready assets, resulting in realistic deformable simulations.

1. Introduction

Surging interest in robotics is amplifying the demand for realistic digital environments, suitable for training robotic agents with physics simulation in the loop. However, constructing such environments remains labor-intensive. Typical 3D scenes, authored, generated or captured from photos, lack the parameters necessary for physics simulation, notably the mechanical properties, Young's modulus (E), Poisson's ratio (ν) and density (ρ), all of which are *spatially-varying* and must be defined *throughout each object's volume* to ensure accurate simulation of real-world behaviors. Accurately assigning such properties manually is difficult to impossible, and measuring real-world objects do not scale with the demand for digital simulation. Recent works (Dagli et al., 2025; Le et al., 2025) propose techniques that learn to predict spatially varying material properties for 3D objects automatically, but are limited in either their accuracy or resolution.

We propose ADAVOMP, a method for predicting accurate spatially-varying mechanical properties (E, ν, ρ) for input 3D shapes using an adaptive structure, improving the resolution, accuracy, and memory efficiency over state-of-the-art. Our method replaces the fixed voxel model of the most accurate prior method, VoMP (Dagli et al., 2025), with a novel sparse transformer encoder-decoder model that learns to generate a unique adaptive structure for every input shape to compactly represent its material distribution. This allows us to operate at the max resolution of 1024^3, compared to 64^3 of prior art (Dagli et al., 2025; Le et al., 2025). We introduce an adaptive structure that uses only a few voxels for constant material regions (e.g. the couch armrests in Fig.1), concentrating the model's predictive capacity in more challenging regions and sharp material boundaries, achieving much finer predictions than VoMP (e.g. GPU density in Fig.1).

To accomplish this, we introduce sparse adaptive voxel trees (SAV) to represent the input shape and to autoregressively generate its material field. Unlike prior art (Xiang et al., 2024; Dagli et al., 2025), we aggregate multi-view visual features of the 3D input into a more efficient *adaptive* structure. We introduce a learned sparse transformer encoder for processing this input while attending across multi-level voxels, and a jointly trained sparse transformer generator, that learns to *autoregressively* output materials as a compact SAV representation. We introduce a generative mechanism that predicts both *structure* (per-voxel "KEEP"/"SUBDIVIDE"/"EMPTY") and *material* values at every level. All models are trained jointly on a dataset auto-labeled with a VLM-based pipeline, similar to prior techniques. In summary, our contributions are as follows:

- A **sparse adaptive voxel (SAV)** representation for 3D

[1]NVIDIA [2]University of Toronto. Correspondence to: Rishit Dagli <rdagli@nvidia.com>.

Proceedings of the 43^{rd} International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026.

Figure 1. **ADAVOMP** generates high-resolution physically accurate volumetric mechanical property fields with detailed parts across 3D representations, enabling their use in building realistic interactive worlds and deformable simulations. We simulate a robot interacting with a high-resolution GPU and the sofa or pillows being stable under gravity in this Gaussian splat + mesh environment. (▶: 01 :13)

shapes and materials, designed for transformer-based processing and generation, and efficient querying (§3).
- An **Adaptive Geometry Transformer** that embeds adaptive DINO feature trees with unified coordinate embeddings and sparse windowed attention (§4.1).
- A **novel Generator design with a autoregressive mechanism** for generating SAVs coarse-to-fine (§4.2).
- A **training formulation** for autoregressive generation, combining multi-scales supervision, teacher forcing and explicit empty space negatives (§4.3).
- Significantly higher resolution and accuracy over state-of-the-art mechanical property prediction methods, advancing simulatable environment authoring (§5).
- Extensive ablations of model design and scale (§C.).

Conflict of Interest Disclosure. The authors are employed by NVIDIA, which leads the development of VoMP, which was among the ones evaluated in this paper.

2. Related Works

To accurately predict dynamic behavior, deformable simulations rely on constitutive (or material) models (e.g. Neo-Hookean, St. Venant Kirchoff), which require parameter fields for Young's modulus, Poisson Ratio, and density (E, ν, ρ). Physically accurate parameters are portable across diverse material models, enabling consistent simulation results; in contrast, methods optimizing for computational speed often require modifying those parameters to mitigate numerical instability (Macklin et al., 2016; Sulsky et al., 1994).

Inverse Physics vs. Static Inference. Material parameters can be obtained via expensive real-world measurement or inverse optimization. Inverse physics methods (Zhang et al., 2025; Huang et al., 2024b; Liu et al., 2025; Cleac'h et al., 2023; Liu et al., 2024a; Lin et al., 2025b) optimize parameters from video or priors but suffer from overfitting, simulator-dependence (Sulsky et al., 1994; Le et al., 2025), and poor scalability. In contrast, feed-forward methods like Dagli et al. (2025) and ours, learn from ground truth material datasets and infer volumetric parameters from static scenes, enabling rapid run-time inference.

Mechanical Property Datasets. Predicting volumetric mechanical properties from shape and appearance alone is difficult for learning-based methods, largely due to limited datasets (Gao et al., 2022; Downs et al., 2022; Chen et al., 2025c) and noisy data (Lin et al., 2018), overfit to a simulator (Mishra, 2024; Xie et al., 2025; Belikov et al., 2015), or coarsely annotated (Ahmed et al., 2025; Slim et al., 2023; Li et al., 2022), or limited to rigid objects (Cao et al., 2025). High-quality physical data remains difficult to collect (ASTM Committee D20, 2022; ASTM Committee E28, 2024; Pai, 2000; Loveday et al., 2004). Our model can be trained with part-segmented 3D assets datasets which have mechanical properties, and thus we reuse the VLM data annotation from prior art (Dagli et al., 2025).

Inferring Materials for Static Scenes. Approaches based on NeRF and Gaussian splats (Mildenhall et al., 2020; Kerbl et al., 2023), including (Zhai et al., 2024; Shuai et al., 2025), optimize feature fields that often focus on the surface regions, and cannot model the internal volume. VLM-based methods (Liu et al., 2024b; Chen et al., 2025a; Lin et al., 2025a) allow single-image inference but are computationally heavy and reliant on external segmentation. Other works annotating 3D data (Cao & Kalogerakis, 2025; Cao et al., 2025; Zhao et al., 2024; 2025; Le et al., 2025; Liu et al., 2024a) often target surface properties, or new shape generation, rather than the volumetric augmentation of existing assets. In contrast to these techniques, our method predicts *volumetric* materials for existing shapes.

Comparison to Feed-Forward Methods. We build on top of VoMP (Dagli et al., 2025) by replacing its fixed-resolution grids with a sparse, adaptive voxel tree (Section 3). Similar to VoMP (Dagli et al., 2025), Pixie (Le et al., 2025) also operates on a fixed resolution grid. This allows us to generate coarse-to-fine predictions, scaling to significantly higher effective resolutions in complex regions. While adaptive feature voxel structures are not new (Takikawa et al., 2021), we make the observation that they are especially well-suited for representing volumetric material distributions, which often contain large homogeneous regions (e.g. metal bedframe). Our model is trained to output the least number of voxels, such that if queried with points within the geometry, it would yield correct mechanical properties. Unlike general spatial data structures (e.g., Octrees, OpenVDB) (Deng et al., 2025; Museth et al., 2013) or adaptive discretization models with fixed multi-resolution input (Choudhury et al., 2026), our sparse representation (§ 3) is autoregressively refined specifically for material prediction rather than geometry. Furthermore, we provide a parameterization of building the structure that is differentiable, allowing us to train with such a representation. There exists some recent work (Deng et al., 2025) taht propose an autoregressive formulation that operates over an octree by serializing into a 1D sequence of discrete tokens, which the model generates autoregressively. In contrast, our method maintains the explicit 3D spatial structure throughout the generation process

3. SAV: Sparse Adaptive Voxels

The foundation of our technique is SAV, a sparse adaptive voxel representation that we use to encode both the input 3D shape and the output spatially varying materials. By efficiently representing geometry and materials in adaptive structures, we can allocate less compute to predict areas of piecewise constant materials, common in everyday objects (the wooden surface, the metal frame), while recursively refining only the fine heterogeneous regions and boundaries, enabling our model to predict material fields at $G^3 = 1024^3$ resolution, much higher than 64^3 for VoMP (Dagli et al., 2025) and Pixie (Le et al., 2025).

Unlike general-purpose spatial data structures such as octrees or OpenVDB (Museth et al., 2013), which subdivide based on geometric criteria or explicit thresholds, SAV is autoregressively learned to optimize for *material prediction*. While VoMP (Dagli et al., 2025) and TRELLIS (Xiang et al., 2025) use sparse voxel representations, they operate at a single fixed resolution, requiring all active voxels to be processed at the finest level, which is a prohibitive cost when scaling to high resolutions for volumetric material fields. In contrast, SAV is both sparse *and* adaptive: it stores voxels at multiple resolution levels simultaneously, allocating finer voxels only where material heterogeneity demands them, while representing homogeneous regions with single coarse voxels regardless of their spatial extent.

3.1. Definitions

Here we define SAV structure and basic properties that we utilize during training of our model. For a bounded 3D domain Ω (e.g. $[-0.5, 0.5)^3$), SAV represents a spatially-varying feature field $\mathcal{F} : \Omega \to \mathbb{R}^d$ using an adaptive voxel tree $\mathcal{T}$ whose leaf voxels form an axis-aligned partition of Ω, but may reside at different resolution levels. Each voxel stores its level $\ell \in \{0, \dots, L_{\max}\}$, where 0 is the finest level, and an integer grid index $\mathbf{i} \in \{0, \dots, G_\ell - 1\}^3$ (exponentiation denotes the cartesian product), where $G_\ell := G/2^\ell$ and $L_{\max} := \log_2 G$. To enable our Transformers to attend across resolutions, we also map each voxel (level ℓ, index $\mathbf{i}$) to its *unified* coordinates:

$$\mathbf{u}_{\ell,\mathbf{i}} := 2^\ell \mathbf{i} \in \{0, \dots, G-1\}^3, \tag{1}$$

where $2^\ell \mathbf{i}$ denotes element-wise scalar multiplication $2^\ell \mathbf{i} = (2^\ell i_x, 2^\ell i_y, 2^\ell i_z)$, mapping each voxel to the finest-resolution grid. Given a voxel with index $\mathbf{i} = (i_x, i_y, i_z)$, we further encode its relative position within the parent voxel using its discrete octant id, $o(\mathbf{i}) \in \{0, \dots, 7\}$:

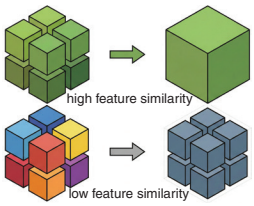
$$o(\mathbf{i}) := (i_x \bmod 2) + 2(i_y \bmod 2) + 4(i_z \bmod 2). \tag{2}$$

Each leaf voxel of level and index $(\ell, \mathbf{i})$ stores a constant feature vector $\mathbf{e}_{\ell,\mathbf{i}} \in \mathbb{R}^d$, inducing a directly queryable piecewise-constant field for spatial queries $\mathbf{x} \in \Omega$, denoted: $\mathcal{T}(\mathbf{x}) := \mathbf{e}_{\ell',\mathbf{i}'}$, where $(\ell', \mathbf{i}')$ is the leaf voxel containing $\mathbf{x}$. We implement querying and construction operations using coordinate-based sparse tensors for ℓ and $\mathbf{i}$, and use a hierarchical hash lookup for fast batched queries. Refer to Section B for details on our memory-efficient GPU implementation.

3.2. Representing the Input Shape

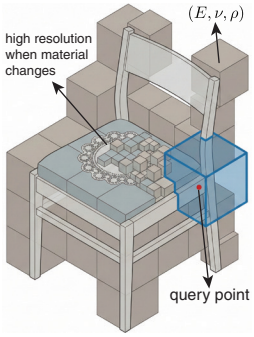
Our goal is to predict material distributions for diverse 3D representations, and we adopt the methodology of VoMP (Dagli et al., 2025), requiring that the 3D input shape be voxelized and renderable from multiple viewpoints, with no other assumptions. First, we discretize the object's occupied volume, normalized to $\Omega \subset [-0.5, 0.5)^3$, into a base grid of resolution $G = 2^{10}$. Then, we aggregate multi-view DINOv3 (Siméoni et al., 2026) patch-token features over this volumetric voxelization, with a few critical differences from prior art. First, to avoid excessive feature averaging that dilutes details, we adopt a depth-attenuated averaging of projected features throughout the voxel structure, in contrast to uniform averaging from prior work (Wang et al., 2023; Dutt et al., 2024; Xiang et al., 2025; Dagli et al., 2025).

Second, after aggregating features in a fixed resolution voxel structure, we progressively voxels with similar features (see inline figure) into an adaptive and more efficient SAV structure $\mathcal{T}^{\text{in}}$, the input to our model. See Section B.2, Section E for details.



3.3. Representing the Material Distribution

We denote the target (ground truth) volumetric material field by $\mathcal{M} : \Omega \to \mathbb{R}^3$, where $\mathcal{M}(\mathbf{x}) = (E(\mathbf{x}), \nu(\mathbf{x}), \rho(\mathbf{x}))$. We represent $\mathcal{M}$ as a SAV material tree $\mathcal{T}^{\mathcal{M}}$ storing material vectors $\mathbf{m}_V \in \mathbb{R}^3$ as features of each voxel $V = (\ell, \mathbf{i})$. We construct $\mathcal{T}^{\mathcal{M}}$ (detailed in Algorithm 2) by: first, aggregating ground truth materials from finest to coarsest, computing the mean and range at each level; second, we traverse coarse-to-fine and subdivide a voxel V only when the material variation within it exceeds a tolerance $\boldsymbol{\tau}$ (computed over its finest-level descendants); otherwise, we keep V and store the descendant mean $\mathbf{m}_V := \frac{1}{|\text{desc}(V)|} \sum_{U \in \text{desc}(V)} \mathbf{m}_U$, where $\text{desc}(V)$ denotes the set of finest-level voxels contained in V. Consequently, partially specified trees remain well-defined *i.e.* missing fine voxels in a region simply return the coarser averaged material for that region, which is a direct mechanism for level-by-level supervision of structure and materials.



4. Learning Adaptive Material Fields

Our goal is to generate a physically accurate volumetric mechanical property field, given an input shape. We first encode the input shape, represented as a SAV $\mathcal{T}^{\text{in}}$ (§3.2), with a trainable Adaptive Geometry Transformer $\mathbf{E}$ (Section 4.1). The resulting latents condition an Adaptive Material Generator $\mathbf{G}$ (Section 4.2), which outputs the final material field, represented as SAV $\mathcal{T}^{\mathcal{M}'}$, at an effective resolution of $G^3 = 1024^3$ without instantiating a dense grid. The $\mathbf{G}$ model operates autoregressively, coarse-to-fine, by predicting both (i) adaptive structure (*i.e.* what spatial regions need high resolutions) and (ii) per-voxel material latents. Both models are trained jointly (Section 4.3), and supervised by ground-truth material trees, also represented as SAV $\mathcal{T}^{\mathcal{M}}$ (Section 3.3).

4.1. Adaptive Geometry Transformer

The input to our encoder $\mathbf{E}$ is a SAV $\mathcal{T}^{\text{in}}$, with aggregated DINOv3 features of the input shape (Section 3.2) as its voxel features $\mathbf{e}_{\ell,\mathbf{i}} \in \mathbb{R}^{d_{\text{in}}}$, $d_{\text{in}} = 1280$. The mixed-level leaf voxels in $\mathcal{T}^{\text{in}}$ form the input sparse token set of $\mathbf{E}$, where each voxel token at level ℓ, index $\mathbf{i}$ is embedded as:

$$\mathbf{h}^0_{\ell,\mathbf{i}} = W_{\text{in}}\, \mathbf{e}_{\ell,\mathbf{i}} + \mathbf{e}^{\text{lvl}}_{\ell}, \tag{3}$$

where W_{in} is a linear projection and $\mathbf{e}^{\text{lvl}}_{\ell}$ is a learned level embedding, ^{0} denotes the initial token embedding (layer 0) before transformer blocks. Additionally, for each token we inject positional information by applying RoPE (Su et al., 2024) on its unified coordinates $\mathbf{u}_{\ell,\mathbf{i}}$ (Eq.1) inside self-attention. We then apply sparse 3D shifted-window self-attention (Liu et al., 2021; 2022; Xiang et al., 2025) in the unified coordinate system, following with a feed-forward network (FFN). Refer to Section F for further details. This yields contextual latents $\mathbf{E}(\mathcal{T}^{\text{in}})$ that serve as conditioning for the Adaptive Material Generator at all generation levels.

4.2. Adaptive Material Generator

Our *autoregressive* transformer model $\mathbf{G}$ generates $\mathcal{T}^{\mathcal{M}'}$ coarse-to-fine, over resolution levels $\ell = L_{\max}, \ldots, 0$. This allows natural test-time compute scaling, yielding well-defined lower-resolution outputs for fewer iterations of $\mathbf{G}$. At each level ℓ we restrict all computation to an explicit sparse candidate set $\mathcal{C}_\ell$ (the refinement frontier), rather than enumerating the full G^3_ℓ grid. For each candidate voxel $(\ell, \mathbf{i}) \in \mathcal{C}_\ell$, $\mathbf{G}$ outputs (i) structure logits over three actions, EMPTY, KEEP, and SUBDIVIDE, and (ii) a latent material vector $\mathbf{z}_{\ell,\mathbf{i}} \in \mathbb{R}^2$ for non-empty voxels. The EMPTY action allows our model to explicitly predict empty space, unlike prior work (Dagli et al., 2025; Lin et al., 2025a; Le et al., 2025; Shuai et al., 2025; Feng et al., 2024).

The transformer model $\mathbf{G}$ is shared across levels, where $\mathbf{G}(\mathcal{C}_\ell)$ yields the candidate set at the next level $\mathcal{C}_{\ell-1}$ along with material latents $\mathbf{z}_{\ell,\mathbf{i}}$ for all KEEP voxels at level ℓ. In addition to its level ℓ and index $\mathbf{i}$, each candidate $(\ell, \mathbf{i}) \in \mathcal{C}_\ell$ also contains its parent's hidden state $\mathbf{h}_{\ell+1,\lfloor \mathbf{i}/2 \rfloor}$ obtained from intermediate layers of prior application of $\mathbf{G}$ (see below). This context from previously subdivided voxels is needed because $\mathcal{C}_\ell$ contains only the refinement frontier, so finer-level candidates would otherwise observe disconnected "holes" wherever coarser KEEP voxels remain unsplit. While the parent necessarily chose SUBDIVIDE for these candidates to exist, this context is essential for spatial coherence. We initialize the coarsest candidate set as $\mathcal{C}_{L_{\max}} = \{(\ell = L_{\max}, \mathbf{i} = (0,0,0))\}$, with $\mathbf{0}$ parent state.

At each level, we construct an initial query embedding for each candidate $(\ell, \mathbf{i}) \in \mathcal{C}_\ell$ by combining its level, octant id (Eq.2), and parent state $\mathbf{h}_{\ell+1,\lfloor \mathbf{i}/2 \rfloor}$:

$$\mathbf{q}_{\ell,\mathbf{i}} = \mathbf{e}^{\text{lvl}}_{\ell} + W_{\text{oct}}\, \mathbf{e}^{\text{oct}}_{o(\mathbf{i})} + W_{\text{par}}\, \mathbf{h}_{\ell+1,\lfloor \mathbf{i}/2 \rfloor}, \tag{4}$$

where W_{oct} and W_{par} are learned linear projections and $\mathbf{e}^{\text{lvl}}_{\ell}$ is a learned level embedding (different from Eq.3).

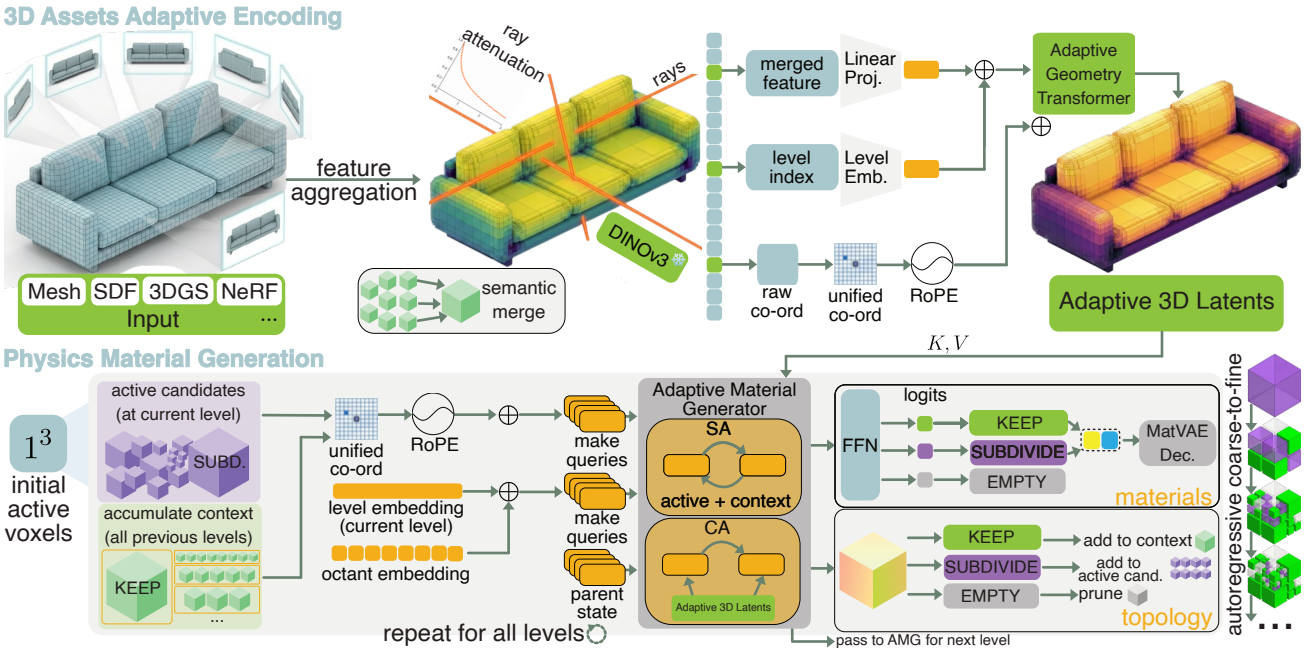
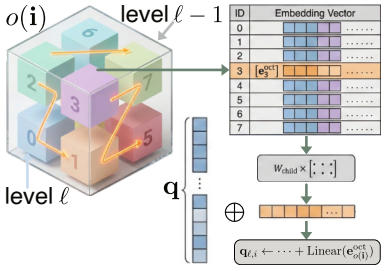
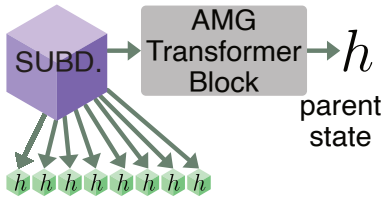


Figure 2. **Method Overview:** input shape is encoded as SAV (top left, §3.2), encoded (top right, §4.1), and processed with our autoregressive Adaptive Material Generator (bottom, §4.2), which is trained (§4.3) to output material field as SAV.

Each candidate also carries its unified coordinate (Eq.1) as its discrete sparse coordinate. We first apply cross-attention from candidates to the input latents $\mathbf{E}(\mathcal{T}^{\text{in}})$ (§4.1), then sparse windowed self-attention among candidates with RoPE (Su et al., 2024) on unified coordinates. See Section 5, Section C for ablation of these choices. From the resulting candidate hidden states $\mathbf{h}_{\ell,\mathbf{i}}$, FFN heads predict the structure action and the 2D material latent $\mathbf{z}_{\ell,\mathbf{i}}$ for every candidate.



If a candidate is predicted as SUBDIVIDE at level $\ell > 0$, we include its eight children in the next candidate set $\mathcal{C}_{\ell-1}$ together with the parent hidden state $\mathbf{h}_{\ell,\mathbf{i}}$. Candidates predicted as EMPTY are discarded, and KEEP voxels remain as leaves of the final material tree. Although these voxels generate no children and thus do not directly appear in finer candidate sets $\mathcal{C}_{\ell'}$ for $\ell' < \ell$, their information is propagated through the hidden state $\mathbf{h}_{\ell,\mathbf{i}}$ (Eq.4), which encodes the broader spatial context, including regions that were kept coarse.



At the finest level $\ell = 0$, refinement terminates and all non-empty voxels are leaves. See Alg.5 for this coarse-to-fine decoding.

MatVAE. To ensure generated properties are physically plausible, we incorporate the frozen MatVAE decoder from VoMP (Dagli et al., 2025), predicting per-voxel latents $\mathbf{z}_{\ell,\mathbf{i}} \in \mathbb{R}^2$ in its latent space. The $\mathbf{z}_{\ell,\mathbf{i}}$ are mapped by MatVAE to (E, ν, ρ), showing improved results (Section C).

4.3. Training

We train $\mathbf{E}$ and $\mathbf{G}$ end-to-end using teacher forcing, jointly supervising structure decisions and node materials. The ground truth $\mathcal{T}^{\mathcal{M}}$ (Section 3.3) stores a material value at every node, where the SUBDIVIDE nodes store descendant means (Section 3). Teacher forcing deterministically fixes the breadth-first refinement schedule by replacing predicted subdivision decisions with ground-truth ones during training. Starting from $\mathcal{C}_{L_{\max}} = \{(L_{\max}, (0,0,0))\}$, we define:

$$\mathcal{C}_{\ell-1} := \bigcup_{(\ell,\mathbf{i})\in\mathcal{C}_\ell:\ s^\star_{\ell,\mathbf{i}}=\text{SUBDIVIDE}} \text{Children}(\ell,\mathbf{i}), \quad (5)$$

for $\ell = L_{\max}, \ldots, 1$, where $s^\star_{\ell,\mathbf{i}}$ denotes the ground-truth structure label in $\mathcal{T}^{\mathcal{M}}$. This construction expands all eight children of every subdividing voxel, ensuring that empty-space children are included as explicit negative candidates. We compute loss across all levels, weighted by $\omega_\ell := \gamma^\ell$,

with $\gamma > 1$, causing larger voxels to contribute more, and optimize the following overall objective:

$$\mathcal{L} = \lambda_{\text{struct}}\,\mathcal{L}_{\text{struct}} + \lambda_{\text{mat}}\,\mathcal{L}_{\text{mat}}, \tag{6}$$

where $\mathcal{L}_{\text{struct}}$ supervises structure actions and $\mathcal{L}_{\text{mat}}$ supervises materials, as detailed in § F.

Supervising Structure. Let $\mathcal{V}^\star_\ell$ denote the grid indices of voxels present at level ℓ in $\mathcal{T}^{\mathcal{M}}$. For a candidate $(\ell, \mathbf{i}) \in \mathcal{C}_\ell$, we define its ground truth structure decision as:

$$s^\star_{\ell,\mathbf{i}} := \begin{cases} \textsc{Empty}, & \mathbf{i} \notin \mathcal{V}^\star_\ell, \\ \textsc{Subdivide}, & \mathbf{i} \in \mathcal{V}^\star_\ell \text{ and } \ell > 0 \text{ and} \\ & \text{Children}(\ell, \mathbf{i}) \cap \mathcal{V}^\star_{\ell-1} \neq \emptyset \\ \textsc{Keep}, & \text{otherwise.} \end{cases} \tag{7}$$

Then, the structure loss is the candidate-count normalized, level-weighted negative log-likelihood over *all* candidates across levels:

$$\mathcal{L}_{\text{struct}} = \frac{1}{\sum_{\ell=0}^{L_{\max}} |\mathcal{C}_\ell|} \sum_{\ell=0}^{L_{\max}} \omega_\ell \sum_{(\ell,\mathbf{i})\in\mathcal{C}_\ell} \Big(-\log p_{\ell,\mathbf{i}}\big(s^\star_{\ell,\mathbf{i}}\big)\Big), \tag{8}$$

where the EMPTY, SUBDIVIDE, KEEP probabilities are computed as $p_{\ell,\mathbf{i}} = \text{softmax}(\mathbf{a}_{\ell,\mathbf{i}})$ over the structure latents $\mathbf{a}_{\ell,\mathbf{i}}$ output by $\mathbf{G}$, and $p_{\ell,\mathbf{i}}(s^\star_{\ell,\mathbf{i}})$ denotes selecting the probability of the ground truth label. ω^l is the weight for supervising predictions at level l.

Supervising Materials. For material supervision, we only penalize candidates whose ground-truth label is non-empty,

$$\mathcal{P}_\ell := \{(\ell, \mathbf{i}) \in \mathcal{C}_\ell : s^\star_{\ell,\mathbf{i}} \neq \textsc{Empty}\}, \tag{9}$$

since empty candidates have no well-defined material target. For each $(\ell, \mathbf{i}) \in \mathcal{P}_\ell$, we decode the predicted 2D latent through MatVAE to obtain a normalized triplet $\hat{\mathbf{m}}_{\ell,\mathbf{i}} \in \mathbb{R}^3$ and compare it to the normalized target $\mathbf{m}^\star_{\ell,\mathbf{i}}$ from $\mathcal{T}^{\mathcal{M}}$:

$$\mathcal{L}_{\text{mat}} = \frac{1}{\sum_{\ell=0}^{L_{\max}} |\mathcal{P}_\ell|} \sum_{\ell=0}^{L_{\max}} \omega_\ell \sum_{(\ell,\mathbf{i})\in\mathcal{P}_\ell} \big\|\hat{\mathbf{m}}_{\ell,\mathbf{i}} - \mathbf{m}^\star_{\ell,\mathbf{i}}\big\|^2_{\mathbf{\Lambda}}, \tag{10}$$

where $\|\mathbf{v}\|^2_{\mathbf{\Lambda}} := \mathbf{v}^\top \mathbf{\Lambda} \mathbf{v}$ with $\mathbf{\Lambda} = \text{diag}(\lambda_E, \lambda_\nu, \lambda_\rho)$. See Section F for more details.

5. Experiments and Results

Quantitative and qualitative evaluation against prior art (§5.2) show significant improvements in accuracy and resolution. We further ablate our model size (Section 5.3), showing that the gains are not solely due to increased model parameters. See ▶ video and Section A for additional results, Section C for ablations, Section A.2 for end-to-end evaluation running simulation of meshes and Gaussian splats using our predicted materials.

5.1. Implementation Details

Training and Parallelism. We train our models end-to-end in BF16 mixed precision, and develop and implementation that effectively performs Hybrid Sharded Data Parallelism (HSDP) *i.e.* ZeRO-3 (Rajbhandari et al., 2020)/FSDP-2 (Zhao et al., 2023) + Distributed Data Parallelism (DDP) with sparse tensors and sparse operations based on top of Megatron-FSDP (Shoeybi et al., 2020). All of our models are trained on a machine with 32×A100-80 GB GPUs for 5 days. We present additional details in Section F.

Datasets. Our dataset Geometry with Volumetric Trees (GVT) builds on top of the GVM dataset (Dagli et al., 2025), using the same assets slightly expanded by 61 objects. We follow the same VLM annotation using Qwen2.5-VL 72B (Bai et al., 2025), rendering in Omniverse (NVIDIA, 2019) and Blender (Blender Online Community, 2021). For image features we use DINOv3 ViT-H+/16 (Siméoni et al., 2026). Our material tree and feature tree creation were run on a machine with 128×A100-80GB GPUs for two days each. See Section B for data details. We train on 149.50M input tokens and 1.62B output tokens. We report results on two held-out evaluation sets. GVT-TEST contains the same objects as the GVM test split in (Dagli et al., 2025). We additionally evaluate on GVT-HARD, a curated set of 50 objects by including an object if it contains at least one annotated mesh segment that is present under fine voxelization at 1024^3 but is entirely skipped by a coarse 32^3 voxelization. We share additional details in Section E.

5.2. Quantitative and Qualitative Evaluation

We evalute our performance against best recent methods VoMP (Dagli et al., 2025) and Pixie (Le et al., 2025), as well as other baselines NeRF2Physics (Zhai et al., 2024), PUGS (Shuai et al., 2025) and Phys4DGen (Lin et al., 2025a) in Tb.1. All metrics (§D), also used in (Dagli et al., 2025), show significant improvement over state of the art across all three mechanical properties (E, ν, ρ). Note that our method performs better even if evaluated at a lower effective resolution of 64^3, matching many of the baselines. Because our method has significant resolution improvement, we further evaluate on a harder more detailed GVT-HARD dataset (§5.1) in Tb.2, showing an even larger gap in performance, with significant advantages offered by our ADAVOMP.

From qualitative materials fields (Fig.3, § A), we observe that NeRF2Physics (Zhai et al., 2024) and PUGS (Shuai et al., 2025) have highly noisy estimates, Phys4DGen (Lin et al., 2025a) mislabels segments and is unable to segment out complex objects, and Pixie (Le et al., 2025) consistently predicts softer materials and fails on complex objects. VoMP (Dagli et al., 2025) can accurately predict volumetric materials, but for high-resolution objects, VoMP completely

Table 1. **Mechanical Property Estimates** of our method significantly outperform the baselines on all metrics and marginally outperforms the baseline even with low test-time compute (64^3). Per-voxel error rate is first computed per object, then averaged across all objects in the test set to avoid weighing some objects more. Global voxel-level normalization yields similar results (Table 8).

Method	Young's Modulus Pa (E)		Poisson's Ratio (ν)		Density $\frac{kg}{m^3}$ (ρ)	
	ALDE (↓)	ALRE (↓)	ADE (↓)	ARE (↓)	ADE (↓)	ARE (↓)
Evaluation at 64^3 resolution.						
NeRF2Physics (Zhai et al., 2024)	2.8000 (±1.05)	0.1346 (±0.05)	-	-	1432.0343 (±964.88)	1.0365 (±0.63)
PUGS (Shuai et al., 2025)	3.3942 (±1.72)	0.1688 (±0.10)	-	-	3568.2150 (±2839.13)	3.2429 (±3.56)
Phys4DGen* (Lin et al., 2025a)	4.8967 (±3.17)	0.2227 (±0.14)	0.0407 (±0.04)	0.1467 (±0.18)	1865.5673 (±2176.90)	1.4394 (±2.35)
Pixie (Le et al., 2025)	0.3986 (±0.30)	0.0446 (±0.04)	0.0259 (±0.01)	0.0869 (±0.03)	141.7812 (±163.40)	0.0917 (±0.07)
VoMP (Dagli et al., 2025)	0.3793 (±0.29)	0.0409 (±0.04)	0.0241 (±0.01)	0.0818 (±0.03)	142.6949 (±166.90)	0.0921 (±0.07)
Ours-H (0.6B)	**0.3278** (±0.26)	**0.0340** (±0.03)	**0.0205** (±0.01)	**0.0680** (±0.03)	**127.3125** (±150.83)	**0.0842** (±0.07)
Evaluation at 1024^3 resolution.						
NeRF2Physics (Zhai et al., 2024)	4.1273 (±1.71)	0.2064 (±0.09)	-	-	2578.3261 (±1621.75)	1.8734 (±1.06)
PUGS (Shuai et al., 2025)	5.6871 (±2.53)	0.2982 (±0.13)	-	-	6345.9184 (±4012.23)	5.3621 (±4.94)
Phys4DGen* (Lin et al., 2025a)	6.9145 (±4.02)	0.3576 (±0.21)	0.0732 (±0.06)	0.2624 (±0.32)	3187.4207 (±3098.55)	2.9127 (±3.72)
Pixie (Le et al., 2025)	1.2264 (±0.52)	0.1372 (±0.10)	0.0413 (±0.02)	0.1396 (±0.06)	248.6735 (±252.11)	0.1568 (±0.14)
VoMP (Dagli et al., 2025)	1.1371 (±0.36)	0.1226 (±0.08)	0.0289 (±0.01)	0.0965 (±0.04)	191.6284 (±212.77)	0.1216 (±0.09)
Ours-H (0.6B)	**0.8841** (±0.27)	**0.0917** (±0.07)	**0.0215** (±0.01)	**0.0714** (±0.03)	**158.4602** (±176.28)	**0.1048** (±0.08)

Table 2. **GVT-HARD at 1024^3 (object-averaged).** Object-averaged errors on the challenging GVT-HARD subset.

Method	Young's Modulus Pa (E)		Poisson's Ratio (ν)		Density $\frac{kg}{m^3}$ (ρ)	
	ALDE (↓)	ALRE (↓)	ADE (↓)	ARE (↓)	ADE (↓)	ARE (↓)
NeRF2Physics (Zhai et al., 2024)	6.1600 (±2.30)	0.2960 (±0.12)	-	-	3718.4285 (±2376.12)	2.7319 (±1.53)
PUGS (Shuai et al., 2025)	9.0500 (±3.80)	0.4500 (±0.18)	-	-	8157.9374 (±5482.55)	7.4836 (±5.38)
Phys4DGen* (Lin et al., 2025a)	12.3100 (±5.60)	0.5600 (±0.26)	0.1082 (±0.09)	0.3900 (±0.48)	5179.6421 (±4291.83)	3.9738 (±4.43)
Pixie (Le et al., 2025)	1.8950 (±1.10)	0.2120 (±0.11)	0.0492 (±0.03)	0.1650 (±0.09)	393.6274 (±359.28)	0.2586 (±0.24)
VoMP (Dagli et al., 2025)	1.6680 (±0.98)	0.1800 (±0.10)	0.0368 (±0.02)	0.1250 (±0.07)	348.1956 (±317.42)	0.2239 (±0.20)
Ours-H (0.6B)	**1.2440** (±0.44)	**0.1290** (±0.09)	**0.0286** (±0.02)	**0.0950** (±0.06)	**241.8735** (±224.18)	**0.1573** (±0.14)

Table 3. **Mass Estimation.** Errors for estimating mass of objects on the ABO-500 (Collins et al., 2022) dataset, the only existing benchmark, approximating the accuracy of our ρ estimates.

Method	ALDE (↓)	ADE (↓)	ARE (↓)	MnRE (↑)
NeRF2Physics (Zhai et al., 2024)	0.736	12.725	1.040	0.564
PUGS (Shuai et al., 2025)	0.661	9.461	0.767	0.576
Phys4DGen* (Lin et al., 2025a)	0.664	9.961	0.825	0.566
Pixie (Le et al., 2025)	0.654	8.231	0.875	0.584
VoMP (Dagli et al., 2025)	0.631	8.433	0.887	0.576
Ours-H (0.6B)	**0.457**	**6.924**	**0.512**	**0.667**

Table 4. **Material Validity.** We report mean values and relative errors (in %) with the closest physically measured material range in Material Triplet Dataset (Dagli et al., 2025).

Method	$\log(E)$(↓)	ν(↓)	ρ(↓)
NeRF2Physics (Zhai et al., 2024)	1.62 (±4.96)	–	19.75 (±46.60)
PUGS (Shuai et al., 2025)	1.87 (±4.50)	–	13.24 (±12.63)
Phys4DGen* (Lin et al., 2025a)	1.77 (±8.53)	0.85 (±3.01)	39.49 (±35.47)
Pixie (Le et al., 2025)	11.90 (±17.41)	3.46 (±4.42)	46.58 (±36.35)
VoMP (Dagli et al., 2025)	0.29 (±1.23)	**0.00** (±0.00)	**11.75** (±4.02)
Ours-H (0.6B)	**0.28** (±1.27)	**0.00** (±0.00)	11.78 (±3.92)

Table 5. **Ground-truth SAV compactness.** Ratio of leaf nodes at 64^3 resolution or coarser in the ground-truth material tree to the number of occupied voxels under dense 64^3 voxelization.

Metric	GVT-TEST (166)	Full (1,719)
GT leaves ($\ell \leq 6$)	622,022	6,282,369
64^3 voxels	8,586,819	59,621,729
Overall ratio	7.24%	10.54%
Per-object mean	16.16% (±24.57)	14.67% (±30.01)
Median	4.42%	1.97%

Table 6. **Generated vs. ground-truth structure.** Ratios of aggregated counts on GVT-TEST: leaf nodes at levels $\ell \leq 6$ for trees, and occupied voxels for VoMP's dense 64^3 voxelization.

Comparison	Ratio
GT leaves / Generated leaves	79.28%
GT leaves / VoMP voxels (64^3)	7.24%
Generated leaves / VoMP voxels (64^3)	9.14%

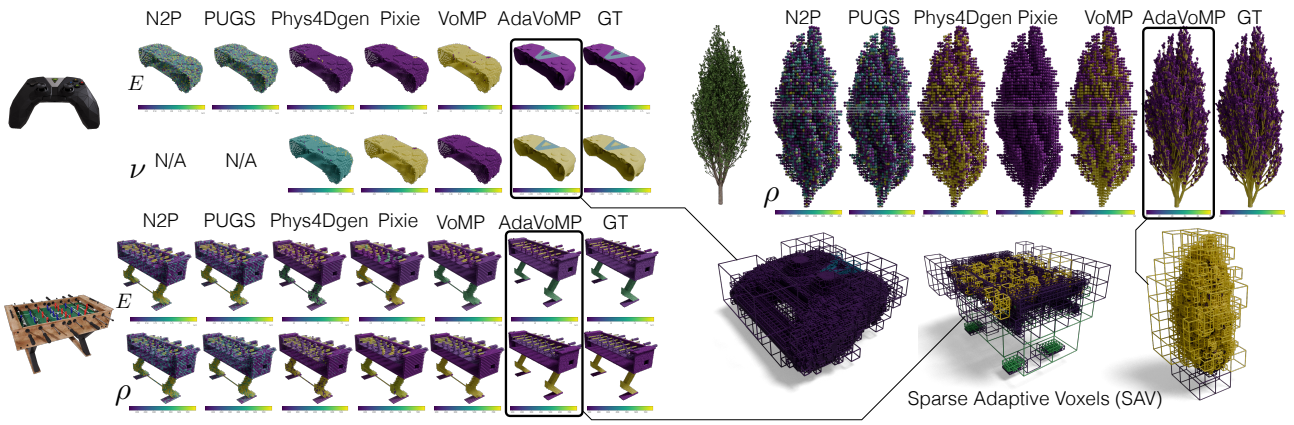


Figure 3. **Qualitative Results:** comparing ADAVOMP material predictions with prior works. These results are generated with our H model with the largest test-time compute. Note: Colorbar scales are different for each algorithm. ▶: 03:18

misses a part of the object due to its low resolution. We demonstrate high-fidelity end-to-end simulations on complex objects in Figures 1 and 5 and Section A.2 (▶: 00:00).

Further, we show on-par or better material validity (whether material is within physically measured material values) against VoMP (Tb.4) on the GVM dataset from VoMP (Dagli et al., 2025), and improved mass estimation on the ABO benchmark in Tb.3.

5.3. Model, Test-Time Compute, and Resolution Scaling

We scale the model to 0.6B parameters and train multiple sizes denoted SMALL (S), BASE (B), BASE+ (B+), LARGE (L), LARGE+ (L+), and HUGE (H) in Figure 4. Apart from these model sizes, we further scale the model in Section A. We find that our B+ model, has similar parameters as VoMP but still outperforms VoMP (Table 7).

5.4. Structure Efficiency

A key advantage of the adaptive SAV representation is its ability to reduce the number of stored voxels compared to a fixed-grid baseline while preserving material fidelity. We quantify this compactness by comparing the number of leaf nodes in ground-truth material trees to the voxel count that would result from a dense 64^3 voxelization, and separately measure how faithfully our generated trees recover the ground-truth structure. Throughout, we report leaf-node counts restricted to levels $\ell \leq 6$, so the finest cells match 64^3 resolution.

Table 5 reports the compactness of the ground-truth SAV representation. On GVT-TEST, ground-truth material trees require only 7.24% of the occupied voxels of a dense 64^3 voxelization (VoMP) when counting leaves at level $\ell \leq 6$ (i.e., 64^3 or coarser) ; on the full dataset the ratio is 10.54%. Per-object statistics reveal substantial variation (median

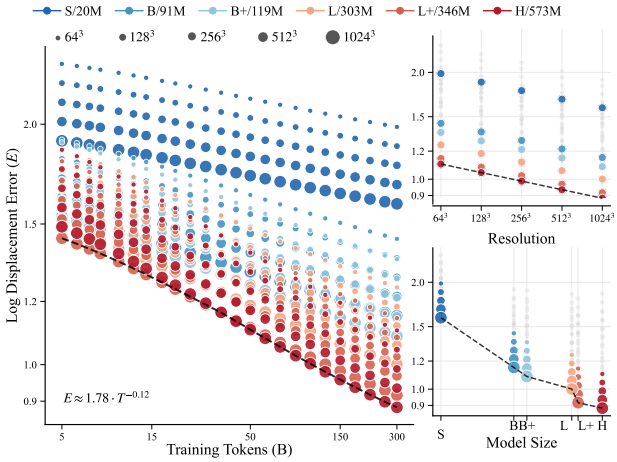


Figure 4. **Scaling Model, Training, and Test-time Compute.** *Left:* Our method shown across three independent axes: training tokens, test-time compute (output resolution), and model size. We show displacement errors for Young's modulus (E) as a function of training tokens. Larger models achieve lower error at a fixed training budget and allocate additional test-time compute (higher resolution) consistently improves accuracy. *Right:* Final training budget and show the error trend as a function of resolution (top) and model size (bottom). A detailed version of this plot is shown in Figure 6.

4.42% on test, 1.97% on full), indicating that many objects are highly compressible while a long tail of complex objects approaches the dense baseline.

Table 6 compares the generated material trees produced by our model to the ground-truth trees and VoMP (Dagli et al., 2025). On GVT-TEST, ground-truth trees contain 79.28% as many leaves as the generated trees, i.e., generated trees use 26.14% more leaves than the oracle structure at levels $\ell \leq 6$. Combining this with the ground-truth compactness ratio of 7.24% (Table 5) implies that generated trees require 9.14% of the occupied voxels of a dense 64^3 voxelization, preserving the compactness advantage after learning.

6. Discussion

As a data-driven method, the accuracy and generalization of our model will improve with more available training data. The high-resolution prediction of mechanical properties by our method enables the approximation of anisotropic materials via multiscale modeling, but still cannot handle truly directional materials whose Young's moduli are spatial-varying tensor fields. Future work can also extend our predictions beyond linear elasticity to include yield strength, shear modulus, and thermal expansion. Our method predicts 'true' material properties that work well for accurate simulators, but it would be useful to adapt to specific, often non-physical parameter scales for approximate, real-time simulators. Lastly, as our approach is designed for static 3D assets, we currently cannot incorporate dynamic physical cues available in video observations. These limitations point to interesting future directions.

7. Conclusion

ADAVOMP predicts mechanical property fields for 3D assets at $16^3\times$ higher resolution than prior works while maintaining memory efficiency. Using surface-level visual appearance to transform 3D assets into volumetric, physically interactable entities, we obviate the need for manual parameter tuning, which is currently the bottleneck to realistic simulation at scale. We hope this work will become a foundational block of physical AI, opening the door to scalable pipelines for generating simulation-ready assets, training robotic agents with physics in the loop, to produce realistic dynamic 3D worlds and to produce realistic interactive worlds.

Acknowledgements

We thank Gilles Daviet for help in setting up some of the simulations. We thank Jean-Francois Lafleche for help with rendering. We thank Beau Perschall, Katherine Cheung for help in using the datasets. We thank Ruchik Thaker for help in releasing the code and data. We thank Andre Pradhana, Anka He Chen, Anita Hu, Charles Loop, Clement Fuji Tsang, Francis Williams, Hexu Zhao and Ken Museth for insightful discussions.

Impact Statement

This paper studies conditional generation of volumetric mechanical property fields from geometric and visual cues. A potential positive impact is to reduce the cost of building simulation-ready digital assets by providing a learned prior over physically plausible, spatially varying materials, which may benefit downstream tasks such as simulation and interactive scene generation. A risk with our model like many other models is that it can be misused to create realistic digital content or deepfakes. The risk is misuse of predicted properties in safety-critical decisions without validation. Our outputs are learned estimates and can be wrong under distribution shift, partial observability, or atypical materials; using them as a substitute for measurement, testing, or certified engineering analysis could lead to unsafe designs or incorrect conclusions. We view the method as a tool for accelerating asset preparation and providing initialization for downstream pipelines, not as a replacement for verification.

References

Ahmed, M., Li, X., Prajapati, A., and Elhoseiny, M. 3dcompat200: Language-grounded compositional understanding of parts and materials of 3d shapes. 2025. URL https://arxiv.org/abs/2501.06785.

An, X., Zhao, L., Gong, C., Wang, N., Wang, D., and Yang, J. Sharpose: Sparse high-resolution representation for human pose estimation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38 (2):691–699, Mar. 2024. doi: 10.1609/aaai.v38i2.27826. URL https://ojs.aaai.org/index.php/AAAI/article/view/27826.

ASTM Committee D20. Standard test method for tensile properties of plastics. Astm standard d638, ASTM International, West Conshohocken, PA, 2022. URL https://doi.org/10.1520/D0638-14. doi:10.1520/D0638-14.

ASTM Committee E28. Standard test methods for tension testing of metallic materials. Astm standard e8/e8m, ASTM International, West Conshohocken, PA, 2024. URL https://doi.org/10.1520/E0008_E0008M-22. doi:10.1520/E0008_E0008M-22.

ASTM International. Standard Test Method for Rubber Property—Durometer Hardness. ASTM Standard D2240-15, 2015. URL https://doi.org/10.1520/D2240-15. doi:10.1520/D2240-15.

Aygun, M. and Mac Aodha, O. Saor: Single-view articulated object reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10382–10391, 2024.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization, 2016. URL https://arxiv.org/abs/1607.06450.

Bai, S., Chen, K., Liu, X., Wang, J., Ge, W., Song, S., Dang, K., Wang, P., Wang, S., Tang, J., Zhong, H., Zhu, Y., Yang, M., Li, Z., Wan, J., Wang, P., Ding, W., Fu, Z., Xu, Y., Ye, J., Zhang, X., Xie, T., Cheng, Z., Zhang, H., Yang, Z., Xu, H., and Lin, J. Qwen2.5-vl technical report, 2025. URL https://arxiv.org/abs/2502.13923.

Bai, Z., Li, W., Yang, G., Meng, F., Kang, R., and Dong, Z. A coarse-to-fine framework for point voxel transformer. In *2024 27th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pp. 205–211, 2024. doi: 10.1109/CSCWD61410.2024.10580279.

Barbere, D., Martin, R., Thornton, B., Harris, C., and Thompson, M. Dynamic token hierarchies: Enhancing large language models with a multi-tiered token processing framework. *Authorea Preprints*, 2024.

Belikov, V., Vabishchevich, N., Vabishchevich, P., Katishkov, U., and Mosunova, N. Material property database. *Mathematical Models and Computer Simulations*, 7:95–102, 2015.

Beyer, L., Izmailov, P., Kolesnikov, A., Caron, M., Kornblith, S., Zhai, X., Minderer, M., Tschannen, M., Alabdulmohsin, I., and Pavetic, F. Flexivit: One model for all patch sizes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14496–14506, June 2023.

Bhat, K. S., Seitz, S. M., Popović, J., and Khosla, P. K. Computing the physical parameters of rigid-body motion from video. In Heyden, A., Sparr, G., Nielsen, M., and Johansen, P. (eds.), *Computer Vision — ECCV 2002*, pp. 551–565, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. ISBN 978-3-540-47969-7.

Black, K., Brown, N., Darpinian, J., Dhabalia, K., Driess, D., Esmail, A., Equi, M. R., Finn, C., Fusai, N., Galliker, M. Y., Ghosh, D., Groom, L., Hausman, K., ichter, b., Jakubczak, S., Jones, T., Ke, L., LeBlanc, D., Levine, S., Li-Bell, A., Mothukuri, M., Nair, S., Pertsch, K., Ren, A. Z., Shi, L. X., Smith, L., Springenberg, J. T., Stachowicz, K., Tanner, J., Vuong, Q., Walke, H., Walling, A., Wang, H., Yu, L., and Zhilinsky, U. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In Lim, J., Song, S., and Park, H.-W. (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 17–40. PMLR, 27–30 Sep 2025. URL https://proceedings.mlr.press/v305/black25a.html.

Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Blender Institute, Amsterdam, 2021. URL http://www.blender.org.

Brubaker, M. A., Sigal, L., and Fleet, D. J. Estimating contact dynamics. In *2009 IEEE 12th International Conference on Computer Vision*, pp. 2389–2396, 2009. doi: 10.1109/ICCV.2009.5459407.

Cao, J. and Kalogerakis, E. Sophy: Learning to generate simulation-ready objects with physical materials, 2025. URL https://arxiv.org/abs/2504.12684.

Cao, Z., Chen, Z., Pan, L., and Liu, Z. Physx-3d: Physical-grounded 3d asset generation. *arXiv preprint arXiv:2507.12465*, 2025.

Chen, B., Jiang, H., Liu, S., Gupta, S., Li, Y., Zhao, H., and Wang, S. Physgen3d: Crafting a miniature interactive world from a single image. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, pp. 6178–6189, June 2025a.

Chen, C., Dou, Z., Wang, C., Huang, Y., Chen, A., Feng, Q., Gu, J., and Liu, L. Vid2sim: Generalizable, video-based reconstruction of appearance, geometry and physics for mesh-free simulation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025b.

Chen, C.-F. R., Fan, Q., and Panda, R. Crossvit: Cross-attention multi-scale vision transformer for image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 357–366, October 2021.

Chen, M., Lin, M., Li, K., Shen, Y., Wu, Y., Chao, F., and Ji, R. Cf-vit: A general coarse-to-fine method for vision transformer. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(6):7042–7052, Jun. 2023. doi: 10.1609/aaai.v37i6.25860. URL https://ojs.aaai.org/index.php/AAAI/article/view/25860.

Chen, Y., Xie, T., Zong, Z., Li, X., Gao, F., Yang, Y., Wu, Y. N., and Jiang, C. Atlas3d: Physically constrained self-supporting text-to-3d for simulation and fabrication, 2024. URL https://arxiv.org/abs/2405.18515.

Chen, Y., Son, H., and Kusari, A. Matpredict: a dataset and benchmark for learning material properties of diverse indoor objects, 2025c. URL https://arxiv.org/abs/2505.13201.

Choudhury, R., Kim, J., Park, J., Yang, E., Jeni, L. A., and Kitani, K. Faster vision transformers with adaptive patches, 2026. URL https://openreview.net/forum?id=SzoowJtd14.

Cleac'h, S. L., Yu, H.-X., Guo, M., Howell, T. A., Gao, R., Wu, J., Manchester, Z., and Schwager, M. Differentiable physics simulation of dynamics-augmented neural objects, 2023. URL https://arxiv.org/abs/2210.09420.

Collins, J., Goel, S., Deng, K., Luthra, A., Xu, L., Gundogdu, E., Zhang, X., Vicente, T. F. Y., Dideriksen, T., Arora, H., Guillaumin, M., and Malik, J. Abo: Dataset and benchmarks for real-world 3d object understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21126–21136, June 2022.

Contributors, S. Spconv: Spatially sparse convolution library. https://github.com/traveller59/spconv, 2022.

Dagli, R., Xiang, D., Modi, V., Loop, C., Tsang, C. F., Chen, A. H., Hu, A., State, G., Levin, D. I. W., and Shugrina, M. Vomp: Predicting volumetric mechanical property fields, 2025. URL https://arxiv.org/abs/2510.22975.

Davis, A., Bouman, K. L., Chen, J. G., Rubinstein, M., Durand, F., and Freeman, W. T. Visual vibrometry: Estimating material properties from small motion in video. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pp. 5335–5343, 2015.

Deng, K., Liu, H.-T. D., Zhu, Y., Sun, X., Shang, C., Bhat, K., Ramanan, D., Zhu, J.-Y., Agrawala, M., and Zhou, T. Efficient autoregressive shape generation via octree-based adaptive tokenization, 2025.

Downs, L., Francis, A., Koenig, N., Kinman, B., Hickman, R., Reymann, K., McHugh, T. B., and Vanhoucke, V. Google scanned objects: A high-quality dataset of 3d scanned household items, 2022. URL https://arxiv.org/abs/2204.11918.

Dutt, N. S., Muralikrishnan, S., and Mitra, N. J. Diffusion 3d features (diff3f): Decorating untextured shapes with distilled semantic features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4494–4504, June 2024.

Fan, Q., You, Q., Han, X., Liu, Y., Tao, Y., Huang, H., He, R., and Yang, H. Vitar: Vision transformer with any resolution, 2024. URL https://arxiv.org/abs/2403.18361.

Feng, Y., Shang, Y., Li, X., Shao, T., Jiang, C., and Yang, Y. Pie-nerf: Physics-based interactive elastodynamics with nerf, 2023.

Feng, Y., Shang, Y., Li, X., Shao, T., Jiang, C., and Yang, Y. Pie-nerf: Physics-based interactive elastodynamics with nerf. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4450–4461, June 2024. URL https://arxiv.org/abs/2311.13099.

frankaemika. franka_description: Official models of franka robotics gmbh robots. https://github.com/frankaemika/franka_description, 2025. GitHub repository, accessed June 2025.

Gao, R., Si, Z., Chang, Y.-Y., Clarke, S., Bohg, J., Fei-Fei, L., Yuan, W., and Wu, J. Objectfolder 2.0: A multisensory object dataset for sim2real transfer, 2022. URL https://arxiv.org/abs/2204.02389.

Ghadai, S., Yeow Lee, X., Balu, A., Sarkar, S., and Krishnamurthy, A. Multi-level 3d cnn for learning multi-scale spatial features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

Goyal, P., Petrov, D., Andrews, S., Ben-Shabat, Y., Liu, H.-T. D., and Kalogerakis, E. Geopard: Geometric pretraining for articulation prediction in 3d shapes. *arXiv preprint arXiv:2504.02747*, 2025.

Guo, M., Wang, B., Ma, P., Zhang, T., Owens, C. E., Gan, C., Tenenbaum, J. B., He, K., and Matusik, W. Physically compatible 3d object modeling from a single image. In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 119260–119282. Curran Associates, Inc., 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/d7af02c8a8e26608199c087f50a21d37-Paper-Conference.pdf.

Havtorn, J. D., Royer, A., Blankevoort, T., and Bejnordi, B. E. Msvit: Dynamic mixed-scale tokenization for vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pp. 838–848, October 2023.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition, 2015. URL https://arxiv.org/abs/1512.03385.

Hu, Y., Cheng, Y., Lu, A., Cao, Z., Wei, D., Liu, J., and Li, Z. Lf-vit: Reducing spatial redundancy in vision transformer for efficient image recognition. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(3):2274–2284, Mar. 2024. doi: 10.1609/aaai.v38i3.28001. URL https://ojs.aaai.org/index.php/AAAI/article/view/28001.

Huang, K., Chitalu, F. M., Lin, H., and Komura, T. Gipc: Fast and stable gauss-newton optimization of ipc barrier energy. *ACM Trans. Graph.*, 43(2), March 2024a. ISSN 0730-0301. doi: 10.1145/3643028. URL https://doi.org/10.1145/3643028.

Huang, K., Lu, X., Lin, H., Komura, T., and Li, M. Stiffgipc: Advancing gpu ipc for stiff affine-deformable simulation. *ACM Trans. Graph.*, 44(3), May 2025. ISSN 0730-0301. doi: 10.1145/3735126. URL https://doi.org/10.1145/3735126.

Huang, T., Zhang, H., Zeng, Y., Zhang, Z., Li, H., Zuo, W., and Lau, R. W. H. Dreamphysics: Learning physics-based 3d dynamics with video diffusion priors, 2024b. URL https://arxiv.org/abs/2406.01476.

Kerbl, B., Kopanas, G., Leimkühler, T., and Drettakis, G. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), July 2023. ISSN 0730-0301. doi: 10.1145/3592433. URL https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

Lang, J., Pai, D. K., and Seidel, H.-P. Scanning large-scale articulated deformations. In *Graphics Interface*, pp. 265–272, 2003.

Le, L., Lucas, R., Wang, C., Chen, C., Jayaraman, D., Eaton, E., and Liu, L. Pixie: Fast and generalizable supervised learning of 3d physics from pixels. 2025. URL https://arxiv.org/abs/2508.17437.

Li, E., Li, T., Luo, H., Chu, J., Duan, L., and Lv, F. Adaptive multi-scale language reinforcement for multimodal named entity recognition. *IEEE Transactions on Multimedia*, 27:5312–5323, 2025a. doi: 10.1109/TMM.2025.3543105.

Li, J., Song, Z., Zhou, S., and Yang, B. Freegave: 3d physics learning from dynamic videos by gaussian velocity. *CVPR*, 2025b.

Li, X., Wang, H., Yi, L., Guibas, L. J., Abbott, A. L., and Song, S. Category-level articulated object pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3706–3715, 2020a.

Li, X., Qiao, Y.-L., Chen, P. Y., Jatavallabhula, K. M., Lin, M., Jiang, C., and Gan, C. Pac-nerf: Physics augmented continuum neural radiance fields for geometry-agnostic system identification, 2023. URL https://arxiv.org/abs/2303.05512.

Li, Y., Lin, T., Yi, K., Bear, D., Yamins, D., Wu, J., Tenenbaum, J., and Torralba, A. Visual grounding of learned physical models. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5927–5936. PMLR, 13–18 Jul 2020b. URL https://proceedings.mlr.press/v119/li20j.html.

Li, Y., Upadhyay, U., Slim, H., Abdelreheem, A., Prajapati, A., Pothigara, S., Wonka, P., and Elhoseiny, M. 3d compat: Composition of materials on parts of 3d things. In Avidan, S., Brostow, G., Cissé, M., Farinella, G. M., and Hassner, T. (eds.), *Computer Vision – ECCV 2022*, pp. 110–127, Cham, 2022. Springer Nature Switzerland. ISBN 978-3-031-20074-8.

Lin, H., Averkiou, M., Kalogerakis, E., Kovacs, B., Ranade, S., Kim, V., Chaudhuri, S., and Bala, K. Learning material-aware local descriptors for 3d shapes. In *2018 International Conference on 3D Vision (3DV)*, pp. 150–159. IEEE, 2018.

Lin, J., Wang, Z., Hou, Y., Tang, Y., and Jiang, M. Phy124: Fast physics-driven 4d content generation from a single image, 2024. URL https://arxiv.org/abs/2409.07179.

Lin, J., Wang, Z., Xu, D., Jiang, S., Gong, Y., and Jiang, M. Phys4dgen: Physics-compliant 4d generation with multi-material composition perception, 2025a. URL https://doi.org/10.1145/3746027.3755647.

Lin, Y., Lin, C., Xu, J., and MU, Y. OmniphysGS: 3d constitutive gaussians for general physics-based dynamics generation. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=9HZtP6I5lv.

Liu, F., Wang, H., Yao, S., Zhang, S., Zhou, J., and Duan, Y. Physics3d: Learning physical properties of 3d gaussians via video diffusion. *arXiv preprint arXiv:2406.04338*, 2024a.

Liu, S., Ren, Z., Gupta, S., and Wang, S. Physgen: Rigid-body physics-grounded image-to-video generation. In *European Conference on Computer Vision*, pp. 360–378. Springer, 2024b.

Liu, Z., Yeh, R. A., Tang, X., Liu, Y., and Agarwala, A. Video frame synthesis using deep voxel flow. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows, 2021. URL https://arxiv.org/abs/2103.14030.

Liu, Z., Hu, H., Lin, Y., Yao, Z., Xie, Z., Wei, Y., Ning, J., Cao, Y., Zhang, Z., Dong, L., Wei, F., and Guo, B. Swin transformer v2: Scaling up capacity and resolution. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11999–12009, 2022. doi: 10.1109/CVPR52688.2022.01170.

Liu, Z., Ye, W., Luximon, Y., Wan, P., and Zhang, D. Unleashing the potential of multi-modal foundation models and video diffusion for 4d dynamic physical scene simulation. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, pp. 11016–11025, June 2025.

Lloyd, J. E. and Pai, D. K. Robotic mapping of friction and roughness for reality-based modeling. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, volume 2, pp. 1884–1890. IEEE, 2001.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Loveday, M. S., Gray, T., and Aegerter, J. Tensile testing of metallic materials: A review. *Final report of the TENSTAND project of work package*, 1, 2004.

Macklin, M., Müller, M., and Chentanez, N. Xpbd: position-based simulation of compliant constrained dynamics. In *Proceedings of the 9th International Conference on Motion in Games*, MIG '16, pp. 49–54, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450345927. doi: 10.1145/2994258.2994272. URL https://doi.org/10.1145/2994258.2994272.

Mezghanni, M., Bodrito, T., Boulkenafed, M., and Ovsjanikov, M. Physical simulation layer for accurate 3d modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13514–13523, June 2022.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

Mishra, A. Latticeml: A data-driven application for predicting the effective young modulus of high temperature graph based architected materials, 2024. URL https://arxiv.org/abs/2404.09470.

Modi, V., Sharp, N., Perel, O., Sueda, S., and Levin, D. I. W. Simplicits: Mesh-free, geometry-agnostic elastic simulation. *ACM Trans. Graph.*, 43(4), July 2024. ISSN 0730-0301. doi: 10.1145/3658184. URL https://doi.org/10.1145/3658184.

Mottaghi, R., Bagherinezhad, H., Rastegari, M., and Farhadi, A. Newtonian scene understanding: Unfolding the dynamics of objects in static images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Museth, K., Lait, J., Johanson, J., Budsberg, J., Henderson, R., Alden, M., Cucka, P., Hill, D., and Pearce, A. Openvdb: an open-source data structure and toolkit for high-resolution volumes. In *ACM SIGGRAPH 2013 Courses*, SIGGRAPH '13, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450323390. doi: 10.1145/2504435.2504454. URL https://doi.org/10.1145/2504435.2504454.

Nawrot, P., Chorowski, J., Lancucki, A., and Ponti, E. M. Efficient transformers with dynamic token pooling. In Rogers, A., Boyd-Graber, J., and Okazaki, N. (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6403–6417, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.353. URL https://aclanthology.org/2023.acl-long.353/.

Ni, J., Chen, Y., Jing, B., Jiang, N., Wang, B., Dai, B., Li, P., Zhu, Y., Zhu, S.-C., and Huang, S. Phyrecon: Physically plausible neural scene reconstruction, 2024. URL https://arxiv.org/abs/2404.16666.

NVIDIA. Isaac Sim. URL https://github.com/isaac-sim/IsaacSim.

NVIDIA. Nvidia unveils omniverse - open, interactive 3d design collaboration platform for multi-tool workflows, 2019. URL https://blogs.nvidia.com/blog/omniverse-collaboration-platform/. NVIDIA Blog.

NVIDIA Corporation. Commercial assets pack. https://docs.omniverse.nvidia.com/usd/latest/usd_content_samples/downloadable_packs.html, 2025a. URL https://docs.omniverse.nvidia.com/usd/latest/usd_content_samples/downloadable_packs.html. Accessed: 2025-06-13.

NVIDIA Corporation. Residential assets pack. https://docs.omniverse.nvidia.com/usd/latest/usd_content_samples/downloadable_packs.html, 2025b. URL https://docs.omniverse.nvidia.com/usd/latest/usd_content_samples/downloadable_packs.html. Accessed: 2025-06-13.

NVIDIA Corporation. Vegetation assets pack. https://docs.omniverse.nvidia.com/usd/latest/usd_content_samples/downloadable_packs.html, 2025c. URL https://docs.omniverse.nvidia.com/usd/latest/usd_content_samples/downloadable_packs.html. Accessed: 2025-06-13.

NVIDIA Corporation. Nvidia physx sdk. https://github.com/NVIDIA-Omniverse/PhysX, 2025d. PhysX SDK (5.x). Accessed: 2026-01-29.

NVIDIA Developer. Simready assets. https://developer.nvidia.com/omniverse/simready-assets, 2025. URL https://developer.nvidia.com/omniverse/simready-assets. Accessed: 2025-06-13.

OpenAI and et al., J. A. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

P, P. J. and Sethi, A. Wavemix: Multi-resolution token mixing for images, 2022. URL https://openreview.net/forum?id=tBoSm4hUWV.

Pai, D. K. Robotics in reality-based modeling. In *Robotics Research: the Ninth International Symposium*, pp. 353–358. Springer, 2000.

Pai, D. K., Doel, K. v. d., James, D. L., Lang, J., Lloyd, J. E., Richmond, J. L., and Yau, S. H. Scanning physical interaction behavior of 3d objects. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pp. 87–96, New York, NY, USA, 2001. Association for Computing Machinery. ISBN 158113374X. doi: 10.1145/383259.383268. URL https://doi.org/10.1145/383259.383268.

Pai, D. K., Lang, J., Lloyd, J., and Woodham, R. J. Acme, a telerobotic active measurement facility. In *Experimental Robotics VI*, pp. 391–400. Springer, 2008.

Pinto, L., Gandhi, D., Han, Y., Park, Y.-L., and Gupta, A. The curious robot: Learning visual representations via physical interactions, 2016. URL https://arxiv.org/abs/1604.01360.

Rajbhandari, S., Rasley, J., Ruwase, O., and He, Y. Zero: Memory optimizations toward training trillion parameter models, 2020. URL https://arxiv.org/abs/1910.02054.

Rao, Y., Zhao, W., Liu, B., Lu, J., Zhou, J., and Hsieh, C.-J. Dynamicvit: Efficient vision transformers with dynamic token sparsification. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 13937–13949. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/747d3443e319a22747fbb873e8b2f9f2-Paper.pdf.

Ren, X., Huang, J., Zeng, X., Museth, K., Fidler, S., and Williams, F. Xcube: Large-scale 3d generative modeling using sparse voxel hierarchies, 2024a. URL https://arxiv.org/abs/2312.03806.

Ren, X., Lu, Y., Liang, H., Wu, Z., Ling, H., Chen, M., Fidler, S., Williams, F., and Huang, J. Scube: Instant large-scale scene reconstruction using voxsplats, 2024b. URL https://arxiv.org/abs/2410.20030.

Ronen, T., Levy, O., and Golbert, A. Vision transformers with mixed-resolution tokenization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 4613–4622, June 2023.

Ryoo, M., Piergiovanni, A., Arnab, A., Dehghani, M., and Angelova, A. Tokenlearner: Adaptive space-time tokenization for videos. In Ranzato, M., Beygelzimer,

A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 12786–12797. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/6a30e32e56fce5cf381895dfe6ca7b6f-Paper.pdf.

Sharp, N. et al. Polyscope, 2019. www.polyscope.run.

Shi, H., Xu, H., Clarke, S., Li, Y., and Wu, J. Robocook: Long-horizon elasto-plastic object manipulation with diverse tools. *arXiv preprint arXiv:2306.14447*, 2023.

Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., and Catanzaro, B. Megatron-lm: Training multi-billion parameter language models using model parallelism, 2020. URL https://arxiv.org/abs/1909.08053.

Shuai, Y., Yu, R., Chen, Y., Jiang, Z., Song, X., Wang, N., Zheng, J., Ma, J., Yang, M., Wang, Z., Ding, W., and Zhao, H. Pugs: Zero-shot physical understanding with gaussian splatting, 2025.

Siméoni, O., Vo, H. V., Seitzer, M., Baldassarre, F., Oquab, M., Jose, C., Khalidov, V., Szafraniec, M., Yi, S. E., Ramamonjisoa, M., Massa, F., HAZIZA, D., Wehrstedt, L., Wang, J., Darcet, T., Moutakanni, T., Sentana, L., Roberts, C., Vedaldi, A., Tolan, J., Brandt, J., Couprie, C., Mairal, J., Jegou, H., Labatut, P., and Bojanowski, P. DINOv3, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=2NlGyqNjns. Featured Certification.

Slim, H., Li, X., Li, Y., Ahmed, M., Ayman, M., Upadhyay, U., Abdelreheem, A., Prajapati, A., Pothigara, S., Wonka, P., et al. 3dcompat++: An improved large-scale 3d vision dataset for compositional recognition. *arXiv preprint arXiv:2310.18511*, 2023.

Song, C., Zhang, J., Li, X., Yang, F., Chen, Y., Xu, Z., Liew, J. H., Guo, X., Liu, F., Feng, J., et al. Magicarticulate: Make your 3d models articulation-ready. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 15998–16007, 2025.

Standley, T., Sener, O., Chen, D., and Savarese, S. image2mass: Estimating the mass of an object from its image. In Levine, S., Vanhoucke, V., and Goldberg, K. (eds.), *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pp. 324–333. PMLR, 13–15 Nov 2017. URL https://proceedings.mlr.press/v78/standley17a.html.

Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2023.127063. URL https://www.sciencedirect.com/science/article/pii/S0925231223011864.

Sulsky, D., Chen, Z., and Schreyer, H. A particle method for history-dependent materials. *Computer Methods in Applied Mechanics and Engineering*, 118(1):179–196, 1994. ISSN 0045-7825. doi: https://doi.org/10.1016/0045-7825(94)90112-0. URL https://www.sciencedirect.com/science/article/pii/0045782594901120.

Takikawa, T., Litalien, J., Yin, K., Kreis, K., Loop, C., Nowrouzezahrai, D., Jacobson, A., McGuire, M., and Fidler, S. Neural geometric level of detail: Real-time rendering with implicit 3d shapes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11358–11367, 2021.

Wang, H., Nie, Y., Ye, Y., Wang, Y., Li, S., Yu, H., Lu, J., and Huang, C. Dynamic-vlm: Simple dynamic visual token compression for videollm. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 20812–20823, October 2025.

Wang, Y., Huang, R., Song, S., Huang, Z., and Huang, G. Not all images are worth 16x16 words: Dynamic transformers for efficient image recognition. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 11960–11973. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/64517d8435994992e682b3e4aa0a0661-Paper.pdf.

Wang, Y., He, X., Peng, S., Lin, H., Bao, H., and Zhou, X. Autorecon: Automated 3d object discovery and reconstruction. In *CVPR*, pp. 21382–21391, June 2023.

Wang, Y., Du, B., Wang, W., and Xu, C. Multi-tailed vision transformer for efficient inference. *Neural Networks*, 174:106235, 2024. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2024.106235. URL https://www.sciencedirect.com/science/article/pii/S089360802400159X.

Wei, S.-T., Wang, R.-H., Zhou, C.-Z., Chen, B., and Wang, P.-S. Octgpt: Octree-based multiscale autoregressive models for 3d shape generation. In *Proceedings of the Special Interest Group on Computer*

Graphics and Interactive Techniques Conference Conference Papers, SIGGRAPH Conference Papers '25, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400715402. doi: 10.1145/3721238.3730601. URL https://doi.org/10.1145/3721238.3730601.

Werby, A., Büchner, M., Röfer, A., Huang, C., Burgard, W., and Valada, A. Articulated object estimation in the wild. In *Conference on Robot Learning (CoRL)*, volume 2, 2025.

World Labs. Marble: A multimodal world model, November 2025. URL https://www.worldlabs.ai/blog/marble-world-model. Published Nov. 12, 2025; accessed 2026-01-04.

Wu, J., Yildirim, I., Lim, J. J., Freeman, B., and Tenenbaum, J. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/d09bf41544a3365a46c9077ebb5e35c3-Paper.pdf.

Wu, J., Lim, J. J., Zhang, H., Tenenbaum, J. B., and Freeman, W. T. Physics 101: Learning physical object properties from unlabeled videos. In *BMVC*, volume 2, pp. 7, 2016.

Wu, J., Lu, E., Kohli, P., Freeman, B., and Tenenbaum, J. Learning to see physics via visual de-animation. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/4c56ff4ce4aaf9573aa5dff913df997a-Paper.pdf.

Xia, H., Lin, Z.-H., Ma, W.-C., and Wang, S. Video2game: Real-time interactive realistic and browser-compatible environment from a single video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4578–4588, June 2024.

Xia, H., Su, E., Memmel, M., Jain, A., Yu, R., Mbiziwo-Tiapo, N., Farhadi, A., Gupta, A., Wang, S., and Ma, W.-C. Drawer: Digital reconstruction and articulation with environment realism. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, pp. 21771–21782, June 2025.

Xiang, D., Modi, V., Dagli, R., Trusty, T., Daviet, G., Chen, A. H., Sharp, N., and Levin, D. I. Freeform: Reduced-order deformable simulation from particle-based skinning eigenmodes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 32475–32484, June 2026.

Xiang, J., Lv, Z., Xu, S., Deng, Y., Wang, R., Zhang, B., Chen, D., Tong, X., and Yang, J. Structured 3d latents for scalable and versatile 3d generation. *arXiv preprint arXiv:2412.01506*, 2024.

Xiang, J., Lv, Z., Xu, S., Deng, Y., Wang, R., Zhang, B., Chen, D., Tong, X., and Yang, J. Structured 3d latents for scalable and versatile 3d generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21469–21480, June 2025. URL https://arxiv.org/abs/2412.01506.

Xie, H., Jia, R., Xia, Y., Li, L., Hu, Y., Xu, J., Sheng, Y., Wang, Y., and Bao, H. An ab initio dataset of size-dependent effective thermal conductivity for advanced technology transistors. *arXiv preprint arXiv:2501.15736*, 2025.

Xu, Z., Wu, J., Zeng, A., Tenenbaum, J. B., and Song, S. Densephysnet: Learning dense physical object representations via multi-step dynamic interactions, 2019. URL https://arxiv.org/abs/1906.03853.

Xue, H., Torralba, A., Tenenbaum, J. B., Yamins, D. L., Li, Y., and Tung, H.-Y. 3d-intphys: Towards more generalized 3d-grounded visual intuitive physics under challenging scenes, 2023. URL https://arxiv.org/abs/2304.11470.

Yan, W., Mnih, V., Faust, A., Zaharia, M., Abbeel, P., and Liu, H. Elastictok: Adaptive tokenization for image and video, 2025. URL https://arxiv.org/abs/2410.08368.

Yang, X., Dagli, R., Zook, A., Hadfield, H., Goyal, A., Birchfield, S., Ramos, F., and Tremblay, J. Robolab: A high-fidelity simulation benchmark for analysis of task generalist policies, 2026. URL https://arxiv.org/abs/2604.09860.

Yang, Y., Jia, B., Zhi, P., and Huang, S. Physcene: Physically interactable 3d scene synthesis for embodied ai. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16262–16272, June 2024.

Yao, S. and Hauser, K. Estimating tactile models of heterogeneous deformable objects in real time. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 12583–12589, 2023. doi: 10.1109/ICRA48891.2023.10160731.

Yildirim, I., Wu, J., Du, Y., and Tenenbaum, J. B. Interpreting dynamic scenes by a physics engine and bottom-up visual cues. *Interpreting dynamic scenes by a physics engine and bottomup visual cues*, 2.

Yu, S., Lin, K., Xiao, A., Duan, J., and Soh, H. Octopi: Object property reasoning with large tactile-language models, 2024. URL https://arxiv.org/abs/2405.02794.

Zhai, A. J., Shen, Y., Chen, E. Y., Wang, G. X., Wang, X., Wang, S., Guan, K., and Wang, S. Physical property understanding from language-embedded feature fields, 2024.

Zhang, B. and Sennrich, R. *Root mean square layer normalization*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Zhang, K., Li, B., Hauser, K., and Li, Y. Adaptigraph: Material-adaptive graph-based neural dynamics for robotic manipulation. *arXiv preprint arXiv:2407.07889*, 2024.

Zhang, T., Yu, H.-X., Wu, R., Feng, B. Y., Zheng, C., Snavely, N., Wu, J., and Freeman, W. T. Physdreamer: Physics-based interaction with 3d objects via video generation. In Leonardis, A., Ricci, E., Roth, S., Russakovsky, O., Sattler, T., and Varol, G. (eds.), *Computer Vision – ECCV 2024*, pp. 388–406, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-72627-9.

Zhao, H., Wang, H., Zhao, X., Fei, H., Wang, H., Long, C., and Zou, H. Efficient physics simulation for 3d scenes via mllm-guided gaussian splatting. *arXiv preprint arXiv:2411.12789*, 2024.

Zhao, H., Wang, H., Zhao, X., Fei, H., Wang, H., Long, C., and Zou, H. Efficient physics simulation for 3d scenes via mllm-guided gaussian splatting. 2025. URL https://arxiv.org/abs/2411.12789.

Zhao, Y., Gu, A., Varma, R., Luo, L., Huang, C.-C., Xu, M., Wright, L., Shojanazeri, H., Ott, M., Shleifer, S., Desmaison, A., Balioglu, C., Damania, P., Nguyen, B., Chauhan, G., Hao, Y., Mathews, A., and Li, S. Pytorch fsdp: Experiences on scaling fully sharded data parallel, 2023. URL https://arxiv.org/abs/2304.11277.

Zhou, Q. and Zhu, Y. Make a long image short: Adaptive token length for vision transformers. In Koutra, D., Plant, C., Gomez Rodriguez, M., Baralis, E., and Bonchi, F. (eds.), *Machine Learning and Knowledge Discovery in Databases: Research Track*, pp. 69–85, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-43415-0.

Supplementary Material for Adaptive Volumetric Mechanical Property Fields Invariant to Resolution

Supplementary Contents

A. Additional Results

A.1. Scaling Experiments

We show full object-averaged scaling results in Tb.7. We complement Figure 4 with a figure scaling the model, training, and test-time compute in Figure 6. We demonstrate experiments on how memory scales in Figure 7 with our framework as we scale model sizes and resolution. We show the dimensionality of generated SAV as we scale the resolution in Figure 8. We show the computational cost for scaling model parameters and resolution in Figure 9.

A.2. End-to-end Examples with Simulation

We qualitatively evaluate ADAVOMP by using it to annotate volumetric mechanical fields for several meshes and 3D Gaussian Splats, and running physics simulation with these spatially varying (E, ν, ρ) values, resulting in realistic simulations without any hand-tweaks.

A.3. Additional Mechanical Property Prediction Results

For completeness, we show voxel (not object) averaged results over regular and hard datasets in Tb.8 and Tb.9, complementing main paper tabulations in Tb.1, Tb.2. These are computed by averaging metrics over all voxels in the dataset.

A.4. Additional Mechanical Property Fields

We show additional mechanical property fields in Figures 12 to 14. We demonstrate additional comparisons with baseline methods in Figures 15 to 19.

B. SAV: Our Sparse Adaptive Volumetric Voxels Backend

We provide additional details about the SAV backend used for training and evaluation. Our goal is to ensure the representation is suitable as a generation target and as conditioning input, while being fast enough to evaluate material properties at high resolutions.

B.1. Representation

Coordinate System and Multi-Resolution Voxels. We operate on a normalized domain $\Omega \subset [-0.5, 0.5)^3$ with finest grid resolution $G = 2^{L_{\max}}$. A level $\ell \in \{0, \dots, L_{\max}\}$ corresponds to voxel side length

$$s_\ell := \frac{2^\ell}{G}, \tag{11}$$

and a level grid size $G_\ell := G/2^\ell$. A level-ℓ voxel is indexed by $\mathbf{i} \in \{0, \dots, G_\ell - 1\}^3$ and corresponds to the axis-aligned cell

$$V_{\ell,\mathbf{i}} := \prod_{\alpha \in \{x,y,z\}} [-0.5 + i_\alpha s_\ell,\ -0.5 + (i_\alpha + 1) s_\ell)\,. \tag{12}$$

Its geometric center is $\mathbf{c}_{\ell,\mathbf{i}} := -0.5 + (\mathbf{i} + 0.5)\, s_\ell$.

Stored Nodes. An adaptive voxel tree stores a sparse subset of voxels at multiple levels. For each level ℓ we store a sparse index set $\mathcal{I}_\ell \subseteq \{0, \dots, G_\ell - 1\}^3$ and associated features $\{\mathbf{f}_{\ell,\mathbf{i}} \in \mathbb{R}^d\}_{\mathbf{i} \in \mathcal{I}_\ell}$. We denote the resulting stored node set by

$$\mathcal{T} := \bigcup_{\ell=0}^{L_{\max}} \{(\ell, \mathbf{i}, \mathbf{f}_{\ell,\mathbf{i}}) : \mathbf{i} \in \mathcal{I}_\ell\}. \tag{13}$$

Finest-Available Query Operator. We interpret SAV as a representation of a function. Given a point $\mathbf{x} \in \Omega$, we define the queried feature as the *finest available* voxel value that covers $\mathbf{x}$:

$$(\ell^*(\mathbf{x}), \mathbf{i}^*(\mathbf{x})) := \arg \min_{\ell \in \{0,\dots,L_{\max}\}} \left\{ \ell \ :\ \left\lfloor \frac{\mathbf{x} + 0.5}{s_\ell} \right\rfloor \in \mathcal{I}_\ell \right\}, \tag{14}$$

$$\mathcal{T}(\mathbf{x}) := \mathbf{f}_{\ell^*(\mathbf{x}),\, \mathbf{i}^*(\mathbf{x})}. \tag{15}$$

When $\mathcal{T}$ is a consistent hierarchy, Equation (15) is equivalent to the usual "leaf voxel" semantics. The operator also remains well-defined for *partial* trees: if fine voxels are missing in a region, queries fall back to a coarser stored voxel and return its region-average feature. This behavior is used for level-wise supervision and test-time compute scaling, since truncating generation yields a valid coarser field.

Material Trees and Averaging under Truncation. For material prediction, we set $\mathcal{F} = \mathcal{M}$ with $d = 3$ and $\mathbf{f}_{\ell,\mathbf{i}} = \mathbf{m}_{\ell,\mathbf{i}} = (E_{\ell,\mathbf{i}}, \nu_{\ell,\mathbf{i}}, \rho_{\ell,\mathbf{i}})$. Our value-range refinement rule stores the descendant mean at coarse voxels when refinement is not triggered. For a voxel $V_{\ell,\mathbf{i}}$ we denote its finest-level descendants by

$$\mathrm{desc}(\ell, \mathbf{i}) := \left\{ \mathbf{j} \in \mathcal{I}_0 \ :\ \lfloor \mathbf{j}/2^\ell \rfloor = \mathbf{i} \right\}, \tag{16}$$

and define the descendant mean

$$\mathbf{m}_{\ell,\mathbf{i}} := \frac{1}{|\mathrm{desc}(\ell, \mathbf{i})|} \sum_{\mathbf{j} \in \mathrm{desc}(\ell,\mathbf{i})} \mathbf{m}_{0,\mathbf{j}}. \tag{17}$$

Thus, if a region is represented only coarsely at inference time, the queried material is the physically meaningful average over that region.

Table 7. **Scaling with model size.** We report object-averaged errors at two query resolutions for all model sizes, using the same evaluation protocol as Table 1. Larger models improve accuracy across (E, ν, ρ). We find that scaling test-time compute ($64^3 \rightarrow 1024^3$) is more effective at larger model sizes.

Model	Young's Modulus Pa (E)		Poisson's Ratio (ν)		Density $\frac{kg}{m^3}$ (ρ)	
	ALDE (↓)	ALRE (↓)	ADE (↓)	ARE (↓)	ADE (↓)	ARE (↓)
Evaluation at 64^3 resolution.						
S	0.5949 (±0.31)	0.0617 (±0.04)	0.0354 (±0.01)	0.1173 (±0.04)	215.7624 (±151.54)	0.1427 (±0.11)
B	0.3828 (±0.24)	0.0397 (±0.03)	0.0232 (±0.01)	0.0768 (±0.04)	136.3824 (±138.64)	0.0902 (±0.06)
B+	0.3625 (±0.23)	0.0376 (±0.04)	0.0223 (±0.01)	0.0741 (±0.03)	133.3584 (±161.00)	0.0882 (±0.07)
L	0.3480 (±0.29)	0.0361 (±0.04)	0.0217 (±0.01)	0.0721 (±0.02)	131.2416 (±154.22)	0.0868 (±0.06)
L+	0.3355 (±0.26)	0.0348 (±0.03)	0.0211 (±0.01)	0.0700 (±0.04)	129.2781 (±164.82)	0.0855 (±0.07)
H	**0.3278** (±0.26)	**0.0340** (±0.03)	**0.0205** (±0.01)	**0.0680** (±0.03)	**127.3125** (±150.83)	**0.0842** (±0.07)
Evaluation at 1024^3 resolution.						
S	1.5898 (±0.32)	0.1649 (±0.10)	0.0439 (±0.01)	0.1457 (±0.03)	299.8296 (±202.80)	0.1983 (±0.10)
B	1.1512 (±0.29)	0.1194 (±0.10)	0.0284 (±0.01)	0.0941 (±0.04)	179.1720 (±228.62)	0.1185 (±0.10)
B+	1.0856 (±0.33)	0.1126 (±0.10)	0.0265 (±0.01)	0.0879 (±0.04)	173.7288 (±176.72)	0.1149 (±0.10)
L	1.0008 (±0.31)	0.1038 (±0.07)	0.0241 (±0.01)	0.0798 (±0.03)	167.2272 (±173.22)	0.1106 (±0.10)
L+	0.9159 (±0.23)	0.0950 (±0.07)	0.0225 (±0.01)	0.0745 (±0.03)	161.7867 (±174.03)	0.1070 (±0.08)
H	**0.8841** (±0.27)	**0.0917** (±0.07)	**0.0215** (±0.01)	**0.0714** (±0.03)	**158.4602** (±176.28)	**0.1048** (±0.08)

Table 8. **Mechanical Property Estimates (voxel-averaged).** of our method significantly outperform the baselines on all metrics and marginally outperforms the baseline even with low test-time compute (64^3). The metrics are averaged across all voxels.

Method	Young's Modulus Pa (E)		Poisson's Ratio (ν)		Density $\frac{kg}{m^3}$ (ρ)	
	ALDE (↓)	ALRE (↓)	ADE (↓)	ARE (↓)	ADE (↓)	ARE (↓)
Evaluation at 64^3 resolution.						
NeRF2Physics (Zhai et al., 2024)	2.5719 (±1.15)	0.4122 (±0.08)	-	-	1354.9458 (±1315.71)	1.1496 (±0.67)
PUGS (Shuai et al., 2025)	3.8619 (±2.01)	0.4512 (±0.11)	-	-	3641.0715 (±3320.78)	4.0413 (±4.16)
Phys4DGen* (Lin et al., 2025a)	5.2977 (±3.36)	0.4825 (±0.14)	0.0394 (±0.05)	0.1425 (±0.21)	1285.9489 (±1981.11)	1.0445 (±2.53)
Pixie (Le et al., 2025)	0.4073 (±0.42)	0.0462 (±0.06)	0.0272 (±0.01)	0.0904 (±0.04)	110.7426 (±294.88)	0.0899 (±0.14)
VoMP (Dagli et al., 2025)	0.3765 (±0.39)	0.0421 (±0.05)	0.0250 (±0.01)	0.0837 (±0.03)	113.3807 (±301.90)	0.0908 (±0.14)
Ours-H (0.6B)	**0.3314** (±0.34)	**0.0342** (±0.04)	**0.0206** (±0.01)	**0.0687** (±0.03)	**96.4381** (±248.62)	**0.0806** (±0.12)
Evaluation at 1024^3 resolution.						
NeRF2Physics (Zhai et al., 2024)	3.9814 (±1.82)	0.6127 (±0.17)	-	-	2548.9372 (±1925.44)	2.0861 (±1.43)
PUGS (Shuai et al., 2025)	6.3189 (±2.97)	0.7421 (±0.22)	-	-	6893.2247 (±4628.35)	6.1443 (±6.21)
Phys4DGen* (Lin et al., 2025a)	7.4136 (±4.28)	0.8317 (±0.32)	0.0789 (±0.08)	0.2912 (±0.41)	2962.5718 (±3254.10)	2.7415 (±3.92)
Pixie (Le et al., 2025)	1.2289 (±0.57)	0.1394 (±0.12)	0.0418 (±0.02)	0.1412 (±0.07)	218.4621 (±268.79)	0.1627 (±0.15)
VoMP (Dagli et al., 2025)	1.1284 (±0.46)	0.1262 (±0.10)	0.0334 (±0.02)	0.1149 (±0.06)	161.9243 (±244.58)	0.1219 (±0.13)
Ours-H (0.6B)	**0.8614** (±0.32)	**0.0889** (±0.09)	**0.0207** (±0.01)	**0.0692** (±0.03)	**124.0773** (±221.37)	**0.1037** (±0.10)

Table 9. **GVT-HARD at 1024^3 (voxel-averaged).** Global voxel-averaged errors on the challenging GVT-HARD subset. Most baselines degrade substantially under voxel averaging, while our gap between voxel and object aggregation remains small.

Method	Young's Modulus Pa (E)		Poisson's Ratio (ν)		Density $\frac{kg}{m^3}$ (ρ)	
	ALDE (↓)	ALRE (↓)	ADE (↓)	ARE (↓)	ADE (↓)	ARE (↓)
NeRF2Physics (Zhai et al., 2024)	5.9300 (±2.70)	0.9500 (±0.22)	-	-	4683.9312 (±2954.17)	3.9045 (±2.12)
PUGS (Shuai et al., 2025)	10.2700 (±4.40)	1.2000 (±0.35)	-	-	10452.8837 (±6890.34)	9.2167 (±6.55)
Phys4DGen* (Lin et al., 2025a)	14.8200 (±6.90)	1.3500 (±0.50)	0.1383 (±0.10)	0.5000 (±0.55)	7421.3764 (±5833.92)	5.6281 (±5.62)
Pixie (Le et al., 2025)	2.8200 (±1.60)	0.3200 (±0.16)	0.0662 (±0.04)	0.2200 (±0.12)	642.9148 (±492.66)	0.3392 (±0.28)
VoMP (Dagli et al., 2025)	2.5480 (±1.45)	0.2850 (±0.14)	0.0568 (±0.03)	0.1900 (±0.10)	571.3873 (±463.21)	0.3184 (±0.26)
Ours-H (0.6B)	**1.2880** (±0.42)	**0.1330** (±0.10)	**0.0300** (±0.02)	**0.1000** (±0.06)	**254.6281** (±237.45)	**0.1651** (±0.15)

Figure 5. **Simulating Gaussian Splats and Meshes at Scale.** We show an elastodynamic simulation of a Gaussian Splat and a mesh scene with objects given mechanical properties generated by ADAVOMP. We find that objects like the sofa and the pillows on the sofa are stable under gravity. Near the center of the scene, we simulate a robot (frankaemika, 2025) which interacts with the fruits on the table producing realistic interactions. We integrate ADAVOMP into RoboLab (Yang et al., 2026) to generate this demo. The Gaussian Splat is generated with Marble (World Labs, 2025) and the robot is controlled by the $\pi_{0.5}$ (Black et al., 2025) Vision-Language-Action model (▶: 04:27).

Training-Only Internal Supervision Nodes. During training, we additionally store internal voxels that are known to be subdivided, solely to define structure supervision (keep vs. subdivide) and per-level losses. These internal nodes do not change the inferred field because queries always return the finest available voxel by Equation (15). We therefore treat this as an auxiliary supervision scaffold, not a distinct inference-time representation.

B.2. Baking DINO Features into SAV

Here we provide details on how multi-view features of the input object are mapped to SAV to be ingested by the Geometry Transformer (§4.1).

We form the conditioning node features by reconstructing multi-view DINOv3 (Siméoni et al., 2026) patch-token features over a volumetric voxelization of the object. Let $\{\mathbf{p}_i\}_{i=1}^{L}$ denote the occupied finest-grid voxel centers and let J denote the set of rendered views. For each view $j \in J$, let $\Pi_j : \mathbb{R}^3 \to [-1,1]^2$ be the camera projection and let $d_{i,j}$ be the camera-space depth of $\mathbf{p}_i$ in view j. Let the DINOv3 patch-token map be $T_j \in \mathbb{R}^{d_{\text{in}} \times n \times n}$ (feature dimension d_{in} on an $n \times n$ patch grid) and let $F_j : [-1,1]^2 \to \mathbb{R}^{d_{\text{in}}}$ denote bilinear sampling of T_j. At our target voxel resolution, many occupied voxels are weakly observed in some views; we therefore model the per-view reconstructed feature as $\tilde{\mathbf{f}}_{i,j} := F_j(\Pi_j(\mathbf{p}_i)) = \mathbf{f}_i^\star + \boldsymbol{\varepsilon}_{i,j}$ with $\mathbb{E}[\boldsymbol{\varepsilon}_{i,j}] = \mathbf{0}$ and depth-dependent variance $\mathbb{E}\|\boldsymbol{\varepsilon}_{i,j}\|_2^2 \propto 1 + \alpha\, \bar{d}_{i,j}$. Using this model, we aggregate features using inverse-depth attenuation,

$$\tilde{w}_{i,j} = \frac{1}{1+\alpha\,\bar{d}_{i,j}}, \quad w_{i,j} = \frac{\tilde{w}_{i,j}}{\sum_{j' \in J} \tilde{w}_{i,j'} + \epsilon}, \\ \mathbf{f}_i = \sum_{j \in J} w_{i,j}\, F_j\big(\Pi_j(\mathbf{p}_i)\big), \tag{18}$$

where $\bar{d}_{i,j} = d_{i,j}/(d_{\max}+\epsilon)$ normalizes depths by $d_{\max} := \max_{i,j} d_{i,j}$ and $\alpha > 0$ controls attenuation strength. The weights are normalized across views so that $\sum_{j \in J} w_{i,j} = 1$ for each voxel i, yielding a depth-weighted average (the case $\alpha = 0$ recovers uniform averaging from prior work (Wang et al., 2023; Dutt et al., 2024; Xiang et al., 2025; Dagli et al., 2025)).

We construct the conditioning tree $\mathcal{T}^{\text{in}}$ by merging feature-homogeneous cells on the SAV grid. For a cell $V_{\ell,\mathbf{i}}$ at level ℓ we sample a subset of finest-level occupied voxel centers $\mathcal{S}_{\ell,\mathbf{i}}^{(K)} \subseteq \{\mathbf{p}_i : \mathbf{p}_i \in V_{\ell,\mathbf{i}}\}$ and reconstruct their features $\{\mathbf{f}_i\}$. We then measure the within-cell feature similarity using the maximum pairwise distance in ℓ_2-normalized feature space,

$$\delta_{\ell,\mathbf{i}} := \max_{i,i' \in \mathcal{S}_{\ell,\mathbf{i}}^{(K)}} \left\| \frac{\mathbf{f}_i}{\|\mathbf{f}_i\|_2} - \frac{\mathbf{f}_{i'}}{\|\mathbf{f}_{i'}\|_2} \right\|_2, \tag{19}$$

and consider the cell uniform if $\delta_{\ell,\mathbf{i}} \le \tau_{\text{feat}}$ for a fixed threshold $\tau_{\text{feat}} > 0$. Uniform cells are stored as leaves with pooled feature given by the mean of sampled (unnormalized) features,

$$\mathbf{e}_{\ell,\mathbf{i}} := \frac{1}{|\mathcal{S}_{\ell,\mathbf{i}}^{(K)}|} \sum_{i \in \mathcal{S}_{\ell,\mathbf{i}}^{(K)}} \mathbf{f}_i, \tag{20}$$

while non-uniform cells are subdivided into occupied children and refined recursively (Algorithm 3).

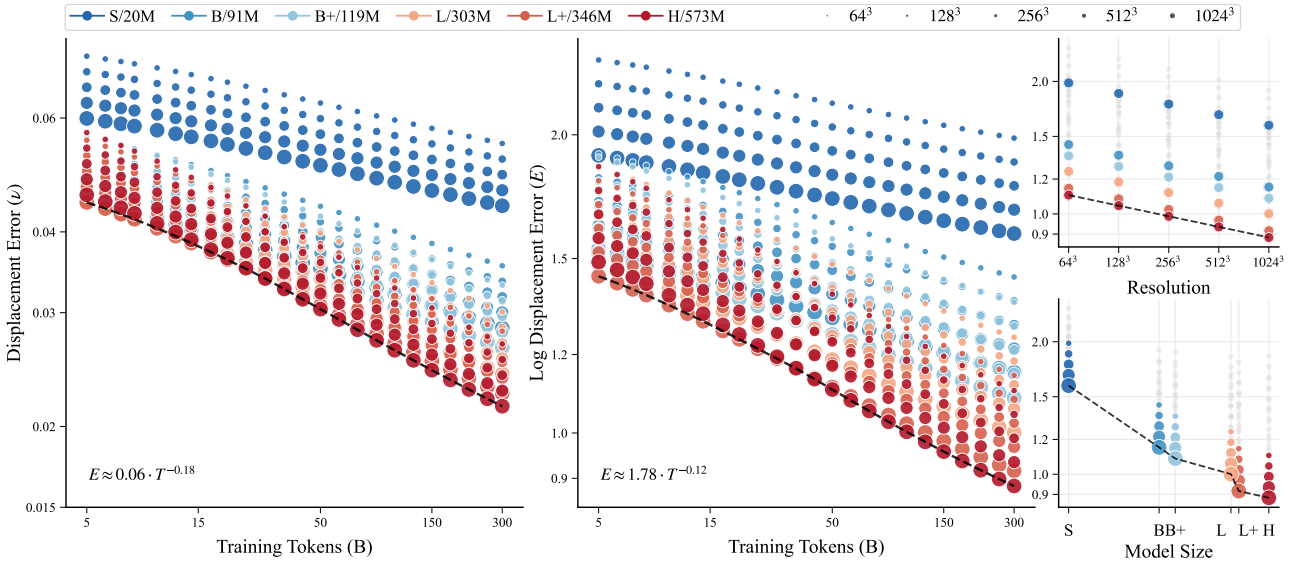


Figure 6. **Scaling Model, Training, and Test-time Compute.** *Left / Center:* We visualize the best runs of our sparse adaptive model across three independent axes: training tokens, test-time compute (output resolution), and model size. We show displacement errors for Poisson's ratio (ν) and Young's modulus (E) as a function of training tokens, showing that larger models achieve lower error at a fixed training budget and that allocating additional test-time compute (higher resolution) consistently improves accuracy. *Right:* We show the final training budget and show the error trend as a function of resolution (top) and model size (bottom).

B.3. Sparse Tensor Backend

Per-Level Sparse Tensors. Each level ℓ is represented as a sparse tensor $\mathcal{S}_\ell := (\mathbf{C}_\ell, \mathbf{F}_\ell)$ with coordinates $\mathbf{C}_\ell \in \mathbb{Z}^{N_\ell \times 4}$ and features $\mathbf{F}_\ell \in \mathbb{R}^{N_\ell \times d}$. Each coordinate row has the form (b, i_x, i_y, i_z) where b is the batch index and $(i_x, i_y, i_z) \in \mathcal{I}_\ell$. For a single tree, $b = 0$ for all rows. For GPU efficiency, we store coordinates as contiguous 32-bit integer tensors with four dimensions (batch id and integer grid indices). The features are stored contiguously in memory.

Hashing for Fast Lookup. For a level grid size G_ℓ, we use a linear spatial hash,

$$h_\ell(i_x, i_y, i_z) := i_x G_\ell^2 + i_y G_\ell + i_z, \tag{21}$$

which is unique on $\{0, \dots, G_\ell - 1\}^3$. In our implementation, we realize membership tests by sorting hashes and applying binary search, yielding $O(\log N_\ell)$ lookup per query.

Batch Flattening across Trees and Levels. Given a batch of trees $\{\mathcal{T}_b\}_{b=0}^{B-1}$, we construct a single batched sparse tensor by concatenating all coordinates/features and writing the batch id into the first coordinate column. Because our encoder conditions on a mixed-level token set, we additionally maintain a per-token level vector $\boldsymbol{\ell} \in \mathbb{Z}^{N_{\text{tot}}}$ aligned with the concatenated rows as we show in Algorithm 1.

Implementation on Top of Sparse Tensor Libraries. We implement sparse voxel tensors using `spconv` (Contributors, 2022) to store $(\mathbf{C}_\ell, \mathbf{F}_\ell)$ and to accelerate common sparse operators. For efficient batching, we maintain an ordering invariant: for each batch element b, all rows with $\mathbf{C}_\ell[:, 0] = b$ form a contiguous slice. This induces a per-batch layout, $\{\text{layout}_\ell[b]\}_{b=0}^{B-1}$ enabling fast extraction of a single item's voxels and efficient broadcast along the batch dimension. When we update features without changing coordinates (a common pattern in neural blocks), we preserve all coordinate metadata and cached index mappings so coordinate-dependent preprocessing is reused instead of recomputed. Finally, we use a spatial cache keyed by scale/stride to reuse expensive coordinate-dependent computations (e.g., sorted hash permutations, window/serialization indices) across repeated operations without changing the stored representation.

B.4. Core Operators

We use four core operators throughout our pipeline: material tree construction (see Algorithm 2), conditioning feature tree construction (see Algorithm 3), batched finest-available point queries (see Algorithm 4), and batch flattening into encoder tokens (see Algorithm 1). We serialize each tree by storing G, the occupied levels, and per-level coordinates and features; this is lossless with respect to Equation (15).

C. Ablations

We provide an in-depth analysis motivating our Adaptive Geometry Transformer and Adaptive Material Generator

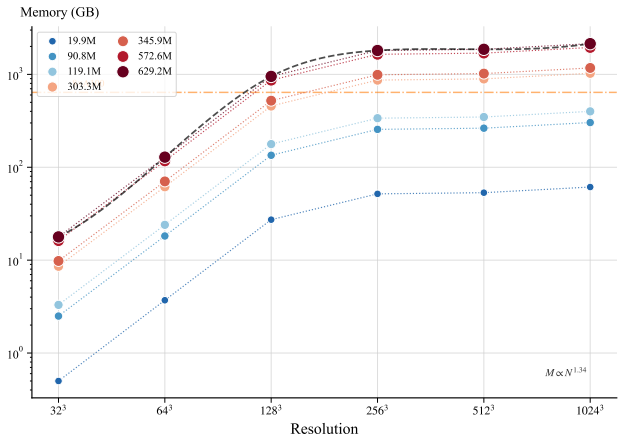


Figure 7. **Memory Scaling.** Peak GPU memory usage versus resolution. We observe a sub-quadratic scaling relationship of $M \propto N^{1.35}$. This efficient scaling allows for the generation of high-resolution 1024^3 volumes within the memory constraints of standard hardware (e.g., 8×A100, dashed orange line). The curves for different model sizes remain parallel, suggesting that the memory overhead from model parameters is independent of the resolution-based scaling.

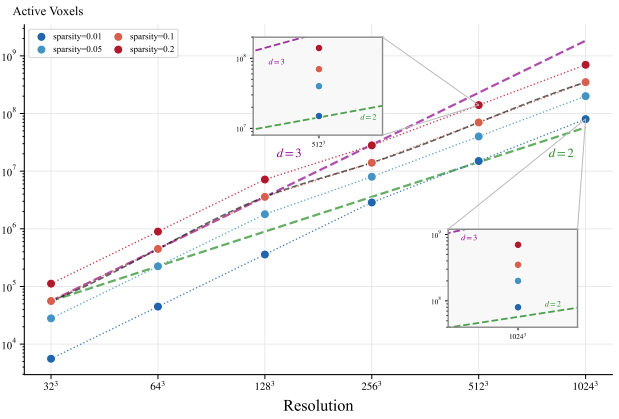


Figure 8. **Effective Dimensionality of Adaptive Geometry.** Active voxel count as a function of resolution for varying sparsity thresholds. The slope of these curves represents the dimension d of the generated geometry. We measure an effective dimensionality of $d \approx 2.48$ for our sparse adaptive volumetric geometry, which falls between surface scaling ($d = 2$) and dense volumetric scaling ($d = 3$). In some cases, SAV can represent a volume more efficiently than representing the same object's surface as a dense voxel grid.

training scheme by ablating each component. Our ablations require changing the hyperparameters for fair comparisons; thus, for each ablation, we tune our hyperparameters within an identical compute budget. We run all the ablations at the B scale (Table 14) which are reported in Table 10. It is not possible to directly compare the results of the Material Gaussian Splats ablation with the baseline because the baseline is trained in a different way with a different architecture. Thus, we do our best to make it comparable to other ablations (Section C.1).

Initialization. We initialize AGT from a pretrained TRELLIS (Xiang et al., 2025) encoder checkpoint (Section F). We ablate this choice by training from scratch and by initializing from a VoMP (Dagli et al., 2025) geometry encoder checkpoint.

Query Embeddings. We ablate the discrete signals used to disambiguate candidate voxels during coarse-to-fine decoding. Removing the level embedding eliminates the explicit level index from the query (Equation (4)), while removing the octant embedding removes the child offset signal (Equations (2) and (4)). We also compare learned level embeddings to a deterministic RoPE-style (Su et al., 2024) level encoding.

Structure and Material Supervision. We ablate the two auxiliary supervision choices used to stabilize coarse-to-fine learning. First, we remove explicit supervision for EMPTY actions by computing $\mathcal{L}_{\text{struct}}$ only on non-empty ground-truth candidates (Equation (8)), which tests whether negative (empty-space) supervision is necessary for reliable structure prediction. Second, we restrict material supervision to the finest level by dropping coarse-level material losses (Equation (10)), testing whether coarse-level averages are needed to regularize long-range material assignments. We note that removing the coarse level material supervision also leads to not being able to scale test-time compute since lower resolutions no longer generate average material values.

Material Parameterization. We ablate the MatVAE constraint by directly regressing the normalized material triplet, removing the learned MatVAE decoder while keeping the same regression loss form (Equation (10)).

C.1. Material Gaussian Splats

Our main method represents the material field as a piecewise-constant adaptive voxel tree, where each leaf voxel stores a constant material vector queried by the tree's spatial lookup. This discretization is effective for scaling to very high effective resolutions, but it ultimately ties the finest representable variation to the finest voxel size. To explore a complementary alternative for capturing sub-voxel material detail, we experimented with a fixed-grid 3D Gaussian Splatting (Kerbl et al., 2023) inspired variant. We keep the training dataset and preprocessing identical to VoMP (Dagli et al., 2025), including fixed-grid voxelization and per-voxel multi-view feature processing. We replace the single per-voxel material latent with a set of Gaussian primitives per voxel.

Representation. We voxelize the object on a fixed 64^3 grid and process up to $L_N = 32{,}768$ occupied voxels per object via stochastic subsampling when the occupied set is larger. Conditioned on the same per-voxel features as VoMP,

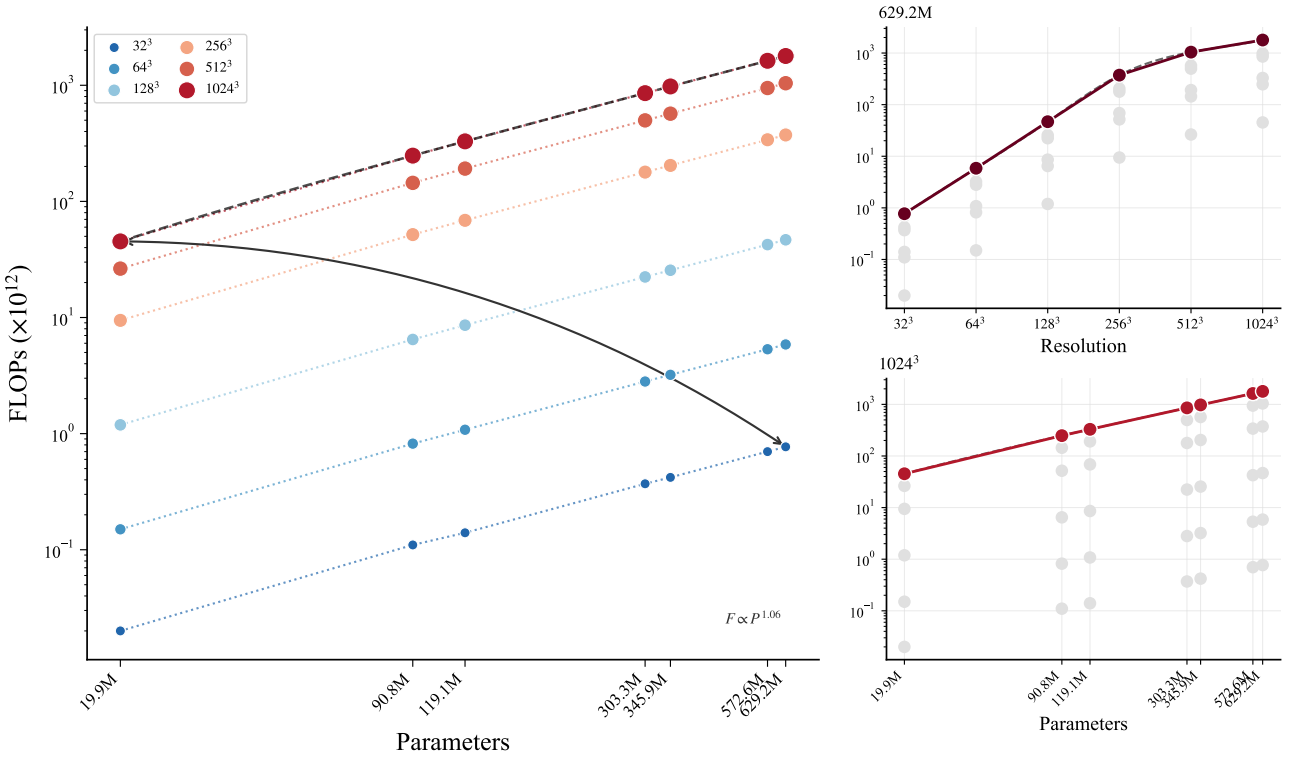


Figure 9. **Parameter versus Resolution Sensitivity. Left:** Total FLOPs as a function of model parameters for various resolutions. **Top Right:** Resolution scaling for the Huge (665M) model. **Bottom Right:** Parameter scaling at fixed 1024^3 resolution. We find that computational cost scales linearly with parameters ($F \propto P^{1.00}$), whereas it scales super-quadratically with resolution ($F \propto N^{2.32}$). The vertical stratification in the left plot confirms that resolution is the dominant driver of compute; increasing model size from 69M to 665M increases cost by roughly $10\times$, whereas increasing resolution from 128^3 to 1024^3 increases cost by two orders of magnitude.

for each occupied voxel V_i with center $\mathbf{c}_i$ and side length h, the decoder predicts $N = 32$ anisotropic 3D Gaussians with parameters $\{(\boldsymbol{\mu}_{i,k}, \boldsymbol{\Sigma}_{i,k}, \alpha_{i,k}, \mathbf{z}_{i,k})\}_{k=1}^{N}$. Here $\boldsymbol{\mu}_{i,k} \in \mathbb{R}^3$ is a center constrained to lie inside V_i, $\boldsymbol{\Sigma}_{i,k}$ is a positive-definite covariance (parameterized by an anisotropic scale and a 3D rotation), $\alpha_{i,k} \in (0, 1)$ is an opacity-like amplitude, and $\mathbf{z}_{i,k} \in \mathbb{R}^2$ is a 2D MatVAE latent code. Unlike appearance-focused splatting, no color is predicted; instead, each Gaussian carries a material latent which is decoded by the frozen MatVAE decoder. This is a direct extension of VoMP's per-voxel latent prediction: a single latent per voxel is replaced by N latents tied to continuous Gaussian supports within the voxel.

Querying the Material Gaussian Splats. Querying $\hat{\mathcal{M}}(\mathbf{x})$ can be viewed as a local "rendering" operator analogous to 3D Gaussian splatting: we evaluate each Gaussian density at a 3D sample point and normalize contributions to obtain mixture weights. Given a query point $\mathbf{x} \in V_i$, we form mixture weights from the Gaussian densities,

$$\tilde{w}_{i,k}(\mathbf{x}) = \alpha_{i,k} \exp\Big(-\tfrac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_{i,k})^\top \boldsymbol{\Sigma}_{i,k}^{-1}(\mathbf{x} - \boldsymbol{\mu}_{i,k})\Big),$$
$$w_{i,k}(\mathbf{x}) = \frac{\tilde{w}_{i,k}(\mathbf{x})}{\sum_{j=1}^{N} \tilde{w}_{i,j}(\mathbf{x})}. \tag{22}$$

We then decode each latent through MatVAE and blend the resulting normalized material triplets,

$$\hat{\mathcal{M}}(\mathbf{x}) = \sum_{k=1}^{N} w_{i,k}(\mathbf{x})\, g_{\text{MATVAE}}(\mathbf{z}_{i,k}). \tag{23}$$

This defines a continuous, locally smooth material field within each voxel while preserving our use of MatVAE to constrain predicted properties.

Training. This ablation follows the supervised fixed-grid recipe of VoMP (Dagli et al., 2025); the only change is to replace the single per-voxel latent with N Gaussian-supported latents inside each occupied voxel. Let $\mathcal{V}$ denote the occupied voxel indices used in an iteration (after subsampling to $|\mathcal{V}| \le L_N$). For each $i \in \mathcal{V}$, we draw a fixed number Q of sample points $\{\mathbf{x}_{i,q}\}_{q=1}^{Q} \subset V_i$ and supervise the

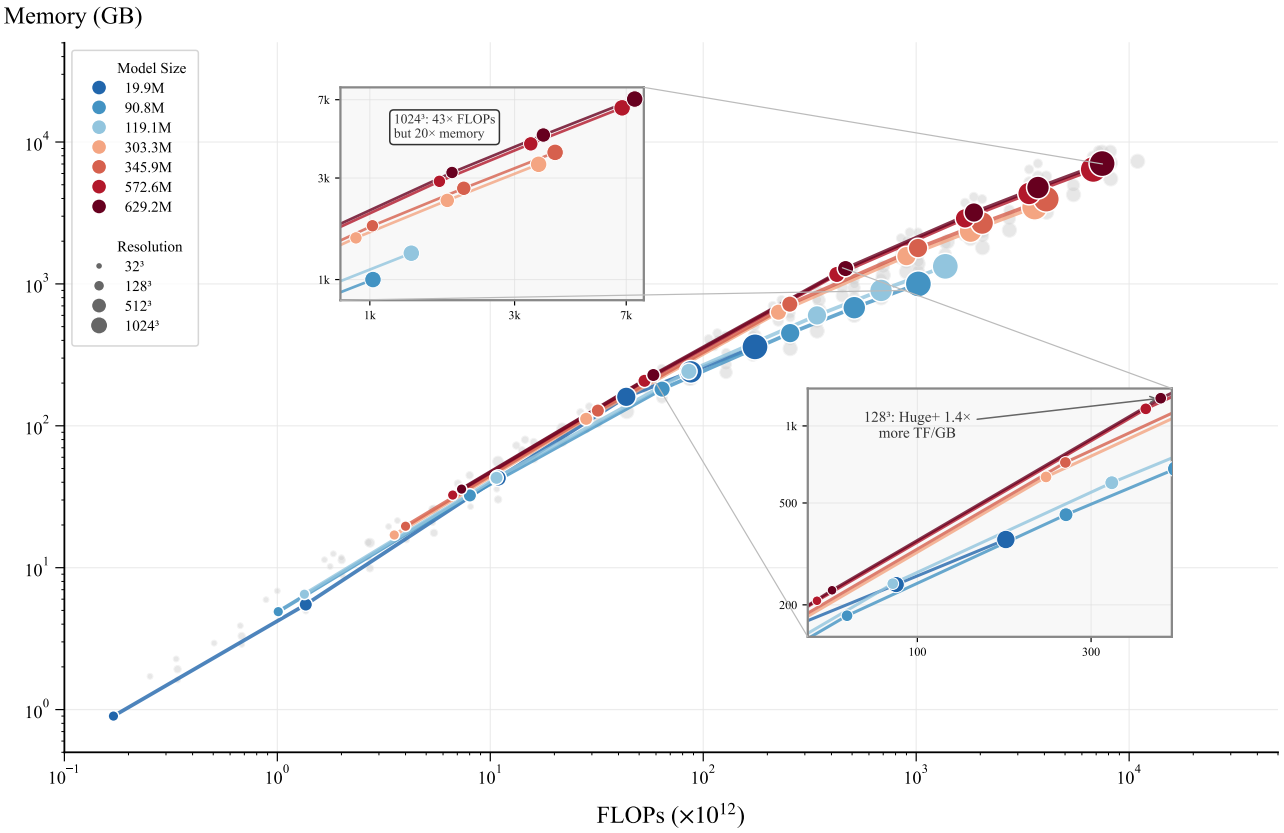


Figure 10. **Compute-memory Pareto Frontier.** Computational cost (FLOPs) versus peak memory usage. The Pareto frontier shows the optimal trade-off between compute and memory. **Insets:** At mid-compute budgets (lower-right inset, 128^3 regime), a model larger than H achieves $1.4\times$ more TFLOPs per GB compared to smaller models. At high-compute budgets (upper-left inset, 1024^3 regime), scaling from the S to larger than H model yields $43\times$ more FLOPs for only $20\times$ more memory.

normalized material field at those points (normalization in Section F). Since $\hat{\mathcal{M}}(\mathbf{x}_{i,q})$ is obtained by the differentiable rendering/query operator in the previous paragraph, training amounts to rendering materials at sampled points and minimizing the point-sampled regression loss

$$\mathcal{L}_{\text{GS}} = \frac{1}{|\mathcal{V}|Q} \sum_{i\in\mathcal{V}} \sum_{q=1}^{Q} \left\| \hat{\mathcal{M}}(\mathbf{x}_{i,q}) - \mathbf{m}^\star(\mathbf{x}_{i,q}) \right\|_{\mathbf{\Lambda}}^2, \quad (24)$$

where $\mathbf{m}^\star(\mathbf{x}) \in \mathbb{R}^3$ denotes the ground-truth normalized material triplet at $\mathbf{x}$ and $\|\cdot\|_{\mathbf{\Lambda}}^2$ is the per-property weighted squared error defined in Equation (10). Empirically, this Gaussian refinement yields only modest gains over the voxel baseline while increasing per-voxel parameterization.

While ablating this model, we choose parameters under the same compute budget as other baselines to serve as a proxy for a fair comparison. This comparison is not perfect since Material Gaussian Splats model is trained with only Data Parallelism which was not possible for the other baselines due to their memory constraints during training. We find that the Material Gaussian Splats are able to outperform the B size, however it is unable to scale well and match performance of our higher model sizes.

D. Metrics

We present an explanation of the metrics we use. We use the same metrics as in VoMP (Dagli et al., 2025).

D.1. Metrics for Mass and Field Estimation

To evaluate the accuracy of predicted scalar quantities such as object mass, as well as continuous scalar fields like density or stiffness, we use several commonly adopted metrics. Let y denote a ground-truth scalar value or voxel-wise field (e.g., density), and $\hat{y}$ its predicted counterpart.

Absolute Difference Error (ADE). The average absolute error between predicted and ground-truth values:

$$\text{ADE} = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i|. \quad (25)$$

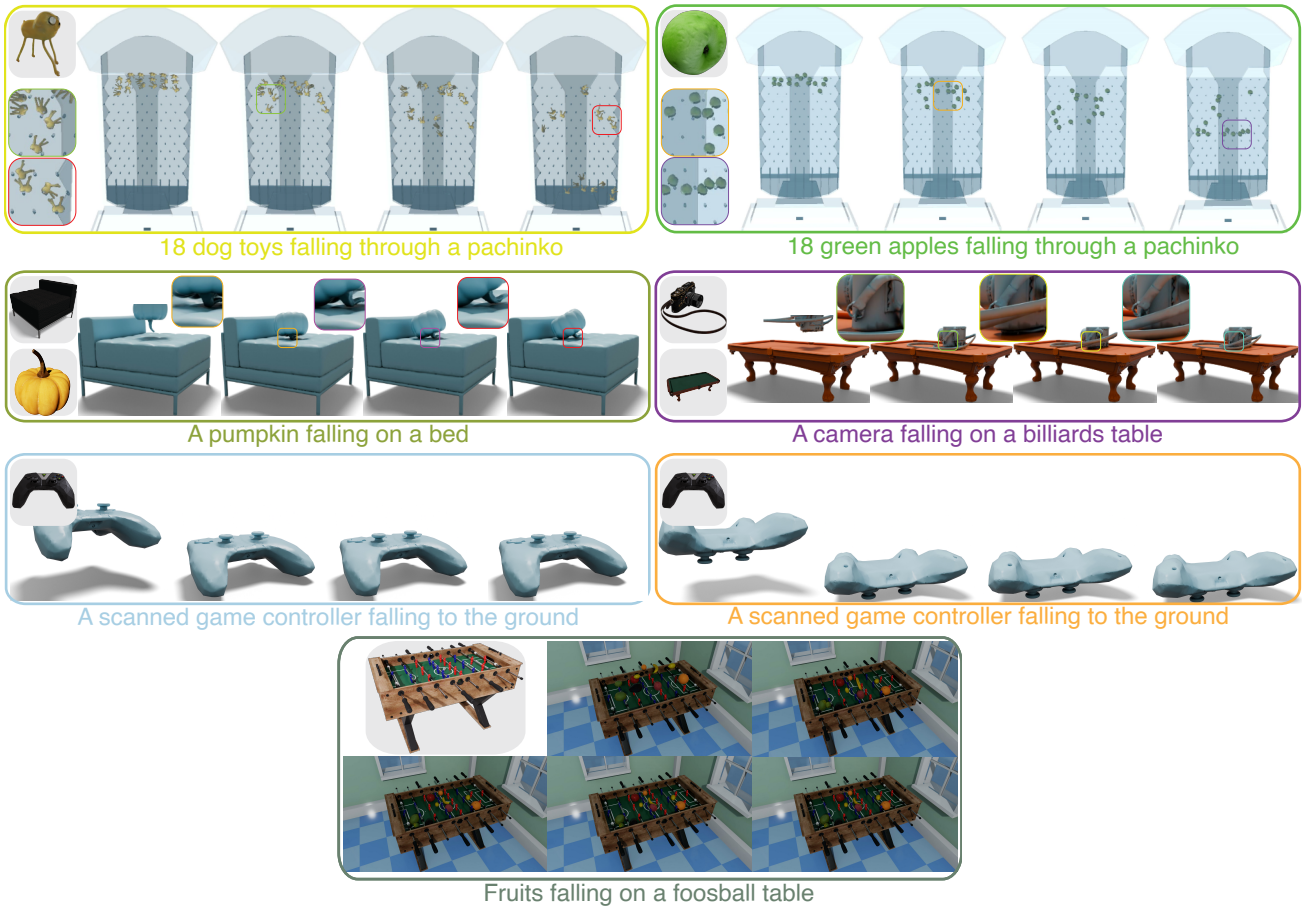


Figure 11. First: We show realistic simulations for 18 Gaussian Splats falling through a pachinko machine mesh using generated properties (▶: 04:44). *Second:* We show realistic simulations for meshes using predicted material values (▶: 04:57). *Third:* In this example, we apply ADAVOMP to this Gaussian Splat model that we captured using a commercial app. Our method converts this model into a simulation-ready asset, which is tetmeshed and simulated with FEM (▶: 04:57). *Fourth:* We show realistic simulations for meshes using predicted material values (▶: 04:44).

Table 10. **Architecture and training ablations at 1024^3.** We evaluate all variants at B scale.

Ablation	Young's Modulus Pa (E)		Poisson's Ratio (ν)		Density $\frac{kg}{m^3}$ (ρ)	
	ALDE (↓)	ALRE (↓)	ADE (↓)	ARE (↓)	ADE (↓)	ARE (↓)
Initialization.						
Scratch initialization	1.3794 (±0.35)	0.1428 (±0.11)	0.0336 (±0.02)	0.1126 (±0.06)	219.4837 (±265.00)	0.1459 (±0.12)
VoMP (Dagli et al., 2025) initialization	1.1916 (±0.30)	0.1236 (±0.10)	0.0292 (±0.01)	0.0968 (±0.04)	185.9342 (±232.00)	0.1214 (±0.10)
Query embeddings.						
w/o level embedding	1.2619 (±0.31)	0.1307 (±0.10)	0.0311 (±0.01)	0.1032 (±0.05)	197.3824 (±240.00)	0.1289 (±0.11)
w/ RoPE (Su et al., 2024) level encoding	1.1884 (±0.30)	0.1227 (±0.10)	0.0290 (±0.01)	0.0961 (±0.04)	184.5173 (±233.00)	0.1207 (±0.10)
w/o octant embedding	1.4682 (±0.40)	0.1579 (±0.12)	0.0674 (±0.04)	0.2386 (±0.18)	419.6341 (±540.00)	0.6852 (±0.55)
Structure and material supervision.						
w/o empty-space supervision	1.3106 (±0.34)	0.1358 (±0.11)	0.0329 (±0.02)	0.1096 (±0.07)	208.1756 (±255.00)	0.1374 (±0.12)
w/ leaf-only material supervision	1.8893 (±0.55)	0.1967 (±0.15)	0.0386 (±0.02)	0.1298 (±0.07)	287.9042 (±350.00)	0.3218 (±0.28)
Material parameterization.						
w/o MatVAE (Dagli et al., 2025)	1.6547 (±0.48)	0.1704 (±0.13)	0.0411 (±0.03)	0.1407 (±0.10)	268.7729 (±330.00)	0.2216 (±0.19)
Material Gaussian Splats.						
Material Gaussian Splats (Section C.1)	**1.1015** (±0.31)	**0.1147** (±0.10)	**0.0273** (±0.01)	**0.0904** (±0.04)	**175.8321** (±222.00)	**0.1168** (±0.10)
Ours-B	1.1512 (±0.29)	0.1194 (±0.10)	0.0284 (±0.01)	0.0941 (±0.04)	179.1720 (±228.62)	0.1185 (±0.10)

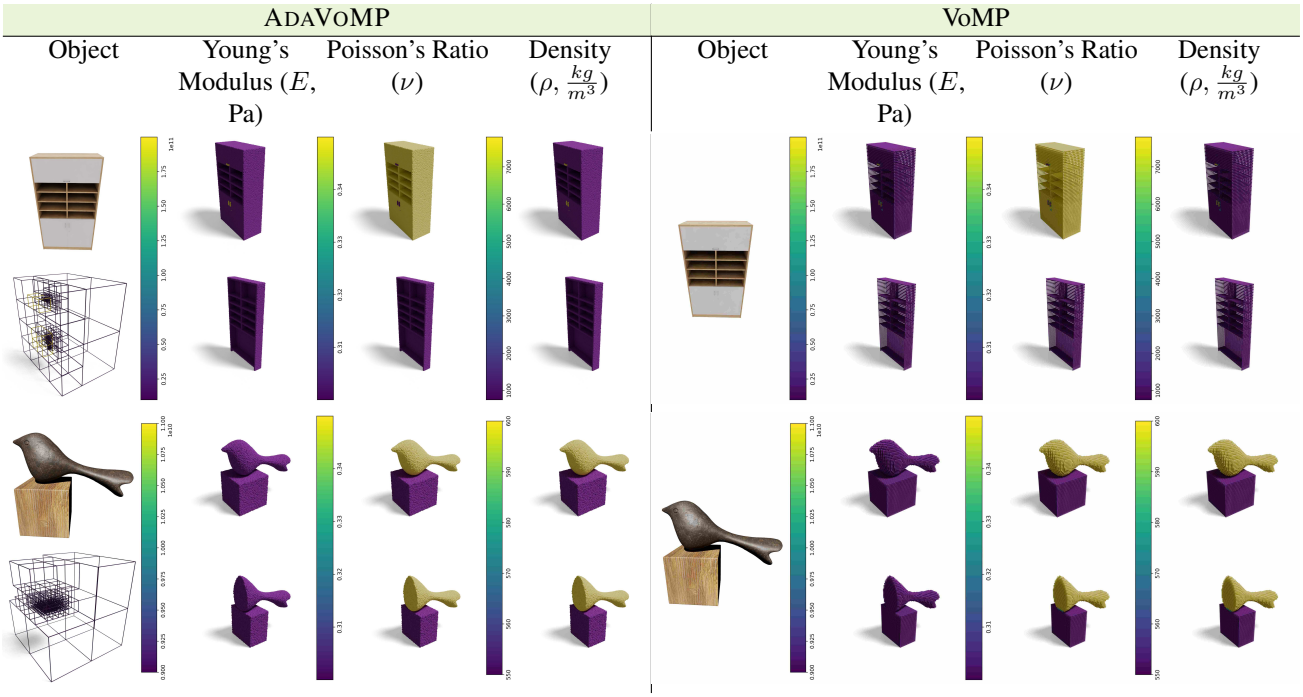


Figure 12. **Inferred Mechanical Property Fields.** We show additional mechanical property fields and slice planes through mechanical property fields estimated by ADAVOMP.

This metric is scale-sensitive and reports the error in physical units (e.g., $\mathrm{kg/m^3}$ for density, kg for mass).

Absolute Log Difference Error (ALDE). The average absolute error in logarithmic space:

$$\mathrm{ALDE} = \frac{1}{N}\sum_{i=1}^{N} |\log y_i - \log \hat{y}_i|. \tag{26}$$

This metric captures multiplicative error and is particularly useful for quantities that vary over several orders of magnitude.

Average Relative Error (ARE). The mean relative deviation between predictions and ground truth:

$$\mathrm{ARE} = \frac{1}{N}\sum_{i=1}^{N} \left|\frac{y_i - \hat{y}_i}{y_i}\right|. \tag{27}$$

This dimensionless metric penalizes over- and under-estimates proportionally, making it appropriate for comparing across varying scales.

Minimum Ratio Error (MnRE). A symmetric and bounded measure of relative accuracy:

$$\mathrm{MnRE} = \frac{1}{N}\sum_{i=1}^{N} \min\left(\frac{y_i}{\hat{y}_i}, \frac{\hat{y}_i}{y_i}\right). \tag{28}$$

This metric ranges from 0 to 1 and is maximized when predictions are perfectly accurate. As suggested in prior work (Standley et al., 2017), MnRE avoids bias toward systematic over- or under-estimation and reduces sensitivity to outliers, making it particularly effective for evaluating physical quantity predictions across heterogeneous samples.

D.2. Metrics to Measure Differences in Mechanical Properties

We evaluate mechanical-property estimates at query points inside the object volume. Let $\{(E_i, \nu_i, \rho_i)\}_{i=1}^{N}$ denote the ground-truth material triplets at the queried locations and let $\{(\hat{E}_i, \hat{\nu}_i, \hat{\rho}_i)\}_{i=1}^{N}$ denote the corresponding predictions.

Relative Error in $\log(E)$. Relative error between predicted and true values of the logarithm of Young's modulus E reported in units of Pa. This captures relative error in material stiffness across several orders of magnitude.

$$\mathrm{ARE}_{\log E} = \frac{1}{N}\sum_{i=1}^{N} \left|\log_{10}\hat{E}_i - \log_{10} E_i\right|. \tag{29}$$

Relative Error in ν. Relative Error in linear space for Poisson's ratio ν, a dimensionless measure of lateral contraction under uniaxial loading.

$$\mathrm{ARE}_{\nu} = \frac{1}{N}\sum_{i=1}^{N} |\hat{\nu}_i - \nu_i|. \tag{30}$$

Algorithm 1 Batch Flattening for Encoder Tokens

Require: Trees $\{\mathcal{T}_b\}_{b=0}^{B-1}$ with per-level sparse tensors $\{(\mathbf{C}_\ell^{(b)}, \mathbf{F}_\ell^{(b)})\}$
Ensure: Batched sparse tensor $(\mathbf{C}_{\text{batch}}, \mathbf{F}_{\text{batch}})$ and per-token levels $\boldsymbol{\ell}$

```
coords ← []
feats ← []
levels ← []
for b = 0, 1, ..., B − 1 do
    for each occupied level ℓ in T_b do
        C ← C_ℓ^(b)
        set C[:, 0] ← b
        Append C to coords
        Append F_ℓ^(b) to feats
        Append a vector of length |C| filled with ℓ to levels
    end for
end for
C_batch ← CAT(coords)
F_batch ← CAT(feats)
ℓ ← CAT(levels)
return (C_batch, F_batch), ℓ
```

Relative Error in ρ. Relative Error between predicted and true values of material density ρ, reported in units of $\mathrm{kg/m^3}$.

$$\mathrm{ARE}_\rho = \frac{1}{N}\sum_{i=1}^{N} |\hat{\rho}_i - \rho_i| \,. \tag{31}$$

Displacement in $\log(E)$. We measure the mean absolute error in $\log_{10}$ space,

$$\mathrm{ADE}_{\log E} = \frac{1}{N}\sum_{i=1}^{N} \left|\log_{10}\hat{E}_i - \log_{10} E_i\right| . \tag{32}$$

Displacement in ν. We measure the mean absolute error of Poisson's ratio in linear space,

$$\mathrm{ADE}_\nu = \frac{1}{N}\sum_{i=1}^{N} |\hat{\nu}_i - \nu_i| \,. \tag{33}$$

Displacement in ρ. We measure the mean absolute error of density in linear space,

$$\mathrm{ADE}_\rho = \frac{1}{N}\sum_{i=1}^{N} |\hat{\rho}_i - \rho_i| \,. \tag{34}$$

E. Dataset Details

This appendix summarizes dataset construction choices that affect the supervision targets and conditioning signals used in our experiments. We refer to our processed dataset as GVT.

E.1. Voxelizing for Training

Assets. We construct GVT from simulation-ready, textured USD assets pooled from multiple collections (NVIDIA Developer, 2025; NVIDIA Corporation, 2025a;b;c). Each asset is assigned a stable instance identifier (SHA-256 hash of its canonical name within its source collection) and a semantic class label provided by source metadata.

Geometry Normalization and Occupancy. All assets are normalized to the coordinate convention of Section 3. We voxelize the occupied volume at the finest grid resolution $G = 1024$ (so $L_{\max} = 10$) and represent geometry by the set of occupied finest-grid indices. We perform volumetric (solid) voxelization by first voxelizing the surface and then filling the interior, yielding occupancy throughout the object volume rather than only on the surface. For robustness, we cap the number of generated voxels per asset at 10^8 during preprocessing; this cap is never active in typical cases.

Per-part Material Supervision. For supervision we require a material triplet (E, ν, ρ) at each occupied voxel (with E in Pa, ν unitless, and ρ in $\mathrm{kg/m^3}$). Each part of the object is assigned a constant material triplet, and each occupied voxel inherits the triplet of its part.

E.2. Material Adaptive Tree for Training

From the voxelized material field at resolution $G = 1024$, we build the supervision tree $\mathcal{T}^{\mathcal{M}}$ using value-range refinement (Algorithm 2). When multiple material samples map to the same voxel, we average them before refinement. We use per-channel merge tolerances

$$\boldsymbol{\tau} = (100.0,\ 0.01,\ 10.0), \tag{35}$$

applied to (E, ν, ρ) in the units above, so that refinement is triggered only at boundaries where descendant ranges exceed these tolerances. Coarser nodes store descendant means, enabling level-wise supervision as described in Section B.

E.3. Feature Adaptive Tree for Training and Inference

Multi-view Rendering. For each asset we render 150 RGB views at 512×512 resolution with transparent background and fixed lighting. Cameras are distributed on a sphere using a low-discrepancy (Hammersley) sequence, with radius 2.1 and horizontal field-of-view $40°$. For each view we record the camera-to-world transform and intrinsics so that features can be lifted consistently to the voxel grid.

DINOv3 Feature Lifting and Aggregation. We use DINOv3 ViT-H+/16 (Siméoni et al., 2026) patch-token features extracted at input resolution 512×512 with patch size 16 (yielding a 32×32 patch grid) and feature dimension

$d_{\text{in}} = 1280$. We apply ImageNet mean/std normalization to each view prior to the DINO forward pass. We lift patch-token features to voxels and aggregate across views as described in Section 4.1, Section B.2 and Equation (18), using inverse-depth attenuation with $\alpha = 2.0$.

Adaptive Feature Tree Hyperparameters. To bound preprocessing cost at $G = 1024$, we cap the number of occupied finest-grid voxels used for feature lifting to 100,000 via stochastic subsampling when the occupied set is larger. We build $\mathcal{T}^{\text{in}}$ using the lazy refinement procedure in Algorithm 3 with start level $\ell_{\text{start}} = 8$, samples per cell $K = 16$, and a maximum of $N_{\max} = 100{,}000$ stored nodes. We use the *max-distance* uniformity test in normalized feature space (Equation (19)) with threshold $\tau_{\text{feat}} = 0.01$.

E.4. Dataset Statistics

We report statistics of GVT after the preprocessing steps in Sections E.1 to E.3 at finest resolution $G = 1024$. GVT uses the same assets as GVM (Dagli et al., 2025) but slightly increases the number of assets to 1,725 high-quality objects. These objects like GVM (Dagli et al., 2025) span four source collections (NVIDIA Developer, 2025; NVIDIA Corporation, 2025a;b;c), dominated by SimReady (59.7%) and Residential (29.3%) assets (Figure 20a). The dataset contains 55 semantic classes with a long-tailed distribution; the most frequent classes are *residential* (29.3%), *shelf* (14.5%), and *container* (11.2%) (Figure 20b).

The volumetric setting induces very large and heavy-tailed occupancy: objects contain 22.5M occupied voxels on median, and the 5th–95th percentile range spans 0.88M–123.0M occupied voxels (Table 11, Figure 20c). At $G = 1024$, this corresponds to ≈ 86.7k input tokens per object (nodes in $\mathcal{T}^{\text{in}}$) and up to 398,112 output tokens per object (decoder candidates $\sum_{\ell=0}^{L_{\max}} |\mathcal{C}_\ell|$ in Section 4.2 under our per-level cap), which are the token counts we train on. This motivates the adaptive discretization and candidate-only computation described in the main text. For sparsity statistics, Figure 20d shows the distribution of adaptive feature-tree sizes (mean 86.7k nodes/object). Figure 21 summarizes how material and feature tree nodes distribute across levels. Reported material-property summary statistics are computed by first averaging each property over the nodes of an object and then taking moments across objects.

Table 11. Summary statistics of GVT after preprocessing at finest resolution $G = 1024$.

Statistic	Value
Dataset	
Objects	1,725
Source collections	4
Semantic classes	55
Voxels	
Total occupied voxels (all objects)	75,266,550,963
Occupied voxels/object (mean)	43,632,783 (±58,967,607)
Occupied voxels/object (median)	22,545,066
Occupied voxels/object (p5–p95)	883,455–123,006,818
Occupied voxels/object (min–max)	8,200–460,452,213
Material tree	
Nodes/object (mean)	936,665 (±6,434,929)
Nodes/object (median)	6,637
Nodes/object (p95)	3,018,837
Nodes/object (max)	115,021,795
Levels/object (mean)	6.6
Total nodes (all objects)	1,615,747,465
Materials	
Young's modulus E (mean)	3.48×10^{10} Pa
Poisson's ratio ν (mean)	0.337
Density ρ (mean)	1949.0 kg/m^3
Feature tree	
Nodes/object (mean)	86,719 (±16,488)
Nodes/object (median)	91,101
Nodes/object (p5–p95)	71,255–96,615
Nodes/object (max)	97,940
Levels/object (mean)	6.7
Total nodes (all objects)	149,503,385
Feature dimension (d_{in})	1280

F. Additional Details on Training

F.1. Network Design

See Section 4 for the definitions of the Adaptive Geometry Transformer (AGT) and Adaptive Material Generator (AMG), including the conditioning tree construction, unified coordinates, and the coarse-to-fine generation procedure. Here we provide the architectural parameterization used in our experiments, together with output-preserving implementation choices that are needed to train at high effective resolutions.

AGT is a sparse 3D Transformer operating on unified coordinates (Equation (1)). We use a pre-norm block with residual connections (He et al., 2015):

$$\begin{aligned} \mathbf{h} &\leftarrow \mathbf{h} + \mathrm{MSA}\big(\mathrm{LN}(\mathbf{h})\big), \\ \mathbf{h} &\leftarrow \mathbf{h} + \mathrm{MLP}\big(\mathrm{LN}(\mathbf{h})\big), \end{aligned} \tag{36}$$

where MSA is multi-head self-attention and MLP is a two-layer feedforward network (MLP ratio 4 with GELU activations). Self-attention uses sparse 3D shifted-window attention (Liu et al., 2021; 2022; Xiang et al., 2025) with RoPE (Su et al., 2024) on unified coordinates. We use width $d_{\text{model}} = 768$ with 12 heads (head dimension 64), and apply per-head RMS normalization (Zhang & Sennrich, 2019) to queries and keys before the dot-product attention. AGT is initialized from a pretrained TRELLIS encoder `slat_enc_swin8_B_64l8` checkpoint (Xiang et al., 2025) and fine-tuned end-to-end.

Table 12. Architectural details of the Adaptive Geometry Transformer (AGT). Shifted-window self-attention uses window size 32 (unified-grid units) and RoPE on unified coordinates; per-head Q/K RMS normalization is applied before dot-product attention. The number of blocks B_{enc} varies by model scale (see Table 14).

Stage	Block
In_proj	Token embedding $\mathbf{h}^0_{\ell,\mathbf{i}}$ (Equation (3))
Stem	$\begin{bmatrix} \text{LayerNorm} \\ \text{QK-RMSNorm} \\ \text{ShiftWinSelfAttn}(12 \times 64) \\ \text{LayerNorm} \\ \text{Linear}(768, 3072) \\ \text{GELU} \\ \text{Linear}(3072, 768) \end{bmatrix} \times B_{\text{enc}}$

AMG details the block structure of the coarse-to-fine generator described in Section 4.2. Each AMG block applies full cross-attention from level-ℓ candidates to AGT latents, followed by sparse shifted-window self-attention among candidates at that level, and uses the same $(d_{\text{model}}, \text{heads}, \text{head dim}, \text{MLP ratio})$ as AGT. We share AMG block weights across refinement levels so that parameter count is independent of $L_{\max}$ while compute scales linearly with the number of levels. The query construction and output heads are summarized in Table 13.

AMG predicts a 2D latent that is decoded by the frozen MatVAE decoder from VoMP (Dagli et al., 2025).

For numerical stability under mixed precision and large token counts, LayerNorm (Ba et al., 2016) is computed in float32 and cast back to the activation dtype. To bound peak memory without changing outputs, we evaluate token-wise operators (LayerNorm (Ba et al., 2016) and the position-wise MLP) in exact chunks when processing very large token sets. Shifted-window attention (Liu et al., 2021; 2022) and RoPE (Su et al., 2024) depend only on the discrete sparse coordinates; since the same coordinate sets are reused across many Transformer blocks and, in AMG, repeatedly across refinement levels, we cache coordinate-dependent quantities such as RoPE $\{\cos, \sin\}$ factors and window-partition index maps and reuse them across blocks. Tables 12 and 13 summarize the block-level architectures.

F.2. Training Recipe

We provide additional details needed to reproduce optimization and normalization, together with the full hyperparameter tables.

We train on normalized material targets to balance gradients across properties. We apply a log-minmax transform to Young's modulus and density and a minmax transform to

Table 13. Architectural details of the Adaptive Material Generator (AMG). Each block applies cross-attention to AGT latents followed by shifted-window self-attention among candidates. AMG blocks are shared across all refinement levels, so the number of blocks B_{dec} is independent of $L_{\max}$ (see Table 14).

Stage	Block
Query	Query embedding $\mathbf{q}_{\ell,\mathbf{i}}$ (Equation (4))
Stem	$\begin{bmatrix} \text{LayerNorm} \\ \text{QK-RMSNorm} \\ \text{CrossAttn}(12 \times 64) \\ \text{LayerNorm} \\ \text{QK-RMSNorm} \\ \text{ShiftWinSelfAttn}(12 \times 64) \\ \text{LayerNorm} \\ \text{Linear}(768, 3072) \\ \text{GELU} \\ \text{Linear}(3072, 768) \end{bmatrix} \times B_{\text{dec}}$
Out_proj	$\text{Linear}(768, 3)$ (structure logits) $\text{Linear}(768, 2)$ (material latent)

Poisson's ratio:

$$E' = \frac{\log_{10}(\max(E, \epsilon)) - \log_{10}(E_{\min})}{\log_{10}(E_{\max}) - \log_{10}(E_{\min})}, \tag{37}$$

$$\nu' = \frac{\nu - \nu_{\min}}{\nu_{\max} - \nu_{\min}}, \tag{38}$$

$$\rho' = \frac{\log_{10}(\max(\rho, \epsilon)) - \log_{10}(\rho_{\min})}{\log_{10}(\rho_{\max}) - \log_{10}(\rho_{\min})}, \tag{39}$$

with a small ϵ to avoid $\log(0)$. We reuse the normalization bounds computed for MatVAE training to ensure the latent decoder and the field generator operate in a consistent normalized space.

We optimize the multi-level objective in Equation (6) with level weights $\omega_\ell = \gamma^\ell$. We use $\lambda_{\text{struct}} = 1$ and report λ_{mat} and the per-property weight matrix $\mathbf{\Lambda} = \text{diag}(\lambda_E, \lambda_\nu, \lambda_\rho)$ in Tables 15 and 16.

We optimize with AdamW (Kingma & Ba, 2017; Loshchilov & Hutter, 2019) for T optimization steps using a linear warmup followed by linear decay. With peak learning rate η_{peak} and warmup length T_{w}, we linearly ramp from $\eta_{\text{peak}}/T_{\text{w}}$ (at step 0) to η_{peak}, and then decay linearly to $\eta_{\text{end}} = 0.01\,\eta_{\text{peak}}$ by step T. We clip the global gradient norm and maintain an exponential moving average of the Adaptive Geometry Transformer and Adaptive Material Generator weights. Training uses bfloat16 mixed precision for the forward pass, while optimizer state is maintained in FP32. We summarize the hyperparameters in Tables 15 and 16. We present details on inference-time decoding, candidate capping, and teacher forcing is provided in Algorithms 5 to 7.

Table 14. Model scale configurations.

Scale	Encoder blocks B_{enc}	Decoder blocks B_{dec}	Params (M)	d_{in}	d_{model}	Heads	Window	d_z
S	1	1	19.9	1280	768	12	32	2
B	6	6	90.8	1280	768	12	32	2
B+	8	8	119.1	1280	768	12	32	2
L	21	21	303.3	1280	768	12	32	2
L+	24	24	345.9	1280	768	12	32	2
H	40	40	572.6	1280	768	12	32	2

F.3. Distributed Training

We train with Hybrid Sharded Data Parallelism (HSDP) *i.e.* ZeRO-3 (Rajbhandari et al., 2020)/FSDP-2 (Zhao et al., 2023) + Distributed Data Parallelism (DDP). We adapt Megatron-LM's Megatron-FSDP (Shoeybi et al., 2020) implementation for our training. We denote an inner group of size S_{shard} with data-parallel replication across R outer replicas. For a world size W, we set $R = W/S_{\text{shard}}$. We summarize the parallelism-related hyperparameters in Tables 15 and 16.

Adaptive trees vary substantially in depth across objects, and many samples terminate early due to no further SUBDIVIDE decisions at coarse levels. Under HSDP, every rank must execute an identical sequence of collectives (parameter gathers and gradient reduce-scatters) in the same order. We therefore enforce a fixed-level schedule during teacher forcing: if a sample has no real candidates at a level, decoding continues with a single dummy candidate for the remaining finer levels, and those dummy levels are masked out of the loss. This keeps collective order aligned across ranks while making it much more computationally efficient to train with diverse data.

We apply global gradient-norm clipping in the sharded setting by first aggregating the squared norm across ranks in the data-parallel group and then scaling local shards accordingly. The total number of input tokens equals the number of nodes in the adaptive feature tree, and the total number of decoder candidates equals the sum of candidate voxels across all levels. Candidate capping bounds the number of candidates processed at a level by selecting a contiguous spatial window, which controls attention memory while preserving locality; pseudo-code is provided in Algorithm 6. For additional memory safety at high resolutions, we compute windowed self-attention in chunks, and we decode MatVAE (Dagli et al., 2025) outputs in chunks when the number of non-empty candidates is large.

G. Additional Implementation Details

G.1. Compute

We report training compute in A100-80GB GPU-days, defined as the number of GPUs multiplied by wall-clock days for an end-to-end training run of AGT+AMG. Table 17 lists the compute required for each model scale in Table 14.

G.2. Simulation and Rendering

G.3. Baselines

Converting Hardness to Young's Modulus. NeRF2Physics (Zhai et al., 2024) does not estimate a numerical value of Young's Modulus, but instead predicts Shore A-Shore D hardness. Thus, to compare our method with NeRF2Physics (Zhai et al., 2024) we convert these Shore hardness values to average Young's Modulus values.

- **Shore A.** For Shore A hardness, we follow (ASTM International, 2015) and use:

$$E_{\text{MPa}} = e^{(S_A \times 0.0235) - 0.6403} \tag{40}$$

  where S_A is the Shore A hardness value and E_{MPa} is Young's modulus in megapascals.
- **Shore D.** For Shore D hardness, we follow (ASTM International, 2015) and use:

$$E_{\text{MPa}} = e^{((S_D + 50) \times 0.0235) - 0.6403} \tag{41}$$

  where S_D is the Shore D hardness value and E_{MPa} is Young's modulus in megapascals.

Point or Voxel Sampling. The baselines NeRF2Physics (Zhai et al., 2024) and PUGS (Shuai et al., 2025) in their methods sample points from the NeRF or Gaussian splat, respectively, and predict mechanical properties at those points. To ensure fair comparisons, we explicitly make these methods work on the same set of points in the object on which our method is evaluated. Pixie (Le et al., 2025) and VoMP (Dagli et al., 2025) work on a fixed lower resolution voxel grid of 64^3. When evaluating Pixie (Le et al., 2025) and VoMP (Dagli et al., 2025) at higher resolutions, we sample voxel centers and

Table 15. Training hyperparameters for S, B, and B+.

Hyperparameter	S	B	B+
Parallelism			
GPUs (W)	16	16	16
ZeRO-3 shard group size (S_{shard})	16	16	16
DDP replica count ($R = W/S_{\text{shard}}$)	1	1	1
Optimization			
Optimizer	AdamW	AdamW	AdamW
AdamW (β_1, β_2)	$(0.9, 0.999)$	$(0.9, 0.999)$	$(0.9, 0.999)$
AdamW ϵ	10^{-8}	10^{-8}	10^{-8}
Weight decay	0.01	0.01	0.01
Training steps (T)	60,000	60,000	60,000
Batch size / GPU	1	1	1
Global batch size ($W\times$ batch/GPU)	16	16	16
Gradient accumulation steps	1	1	1
Gradient clipping (global norm)	1.0	1.0	1.0
EMA decay	0.9999	0.9999	0.9999
Precision (forward)	bf16	bf16	bf16
Learning-rate schedule			
Schedule	linear warmup + linear decay	linear warmup + linear decay	linear warmup + linear decay
Warmup init LR ($\eta_{\text{peak}}/T_{\text{w}}$)	10^{-7}	10^{-7}	10^{-7}
LR peak (η_{peak})	2×10^{-4}	2×10^{-4}	2×10^{-4}
Warmup steps (T_{w})	2,000	2,000	2,000
LR end (η_{end})	2×10^{-6}	2×10^{-6}	2×10^{-6}
Loss			
Structure loss $\mathcal{L}_{\text{struct}}$	3-way cross-entropy	3-way cross-entropy	3-way cross-entropy
Structure loss weight λ_{struct}	1	1	1
Material loss $\mathcal{L}_{\text{mat}}$	ℓ_2 on normalized triplets	ℓ_2 on normalized triplets	ℓ_2 on normalized triplets
Level-weight base γ (for $\omega_\ell = \gamma^\ell$)	1.4	1.4	1.4
Material loss weight λ_{mat}	75	75	75
Per-property weights $\mathbf{\Lambda}$	$(1, 1, 3)$	$(1, 1, 3)$	$(1, 1, 3)$
Sparsity and exact chunking (output-preserving)			
Max candidates / level ($\lvert\mathcal{C}_\ell\rvert_{\max}$)	36,192	36,192	36,192
Window-attention chunk cap (tokens, $N_{\text{chunk}}^{\text{attn}}$)	262,144	262,144	262,144
MatVAE decode chunk ($N_{\text{chunk}}^{\text{MatVAE}}$)	128	128	128

interpolate the mechanical properties at those points from the lower resolution voxel grid.

Implementation details of Baselines. The baseline NeRF2Physics (Zhai et al., 2024) uses `gpt-3.5-turbo` for certain parts of their pipeline. We replace `gpt-3.5-turbo` in their pipeline with a better performing model, GPT-4o (OpenAI & et al., 2024). The baseline Phys4DGen (Lin et al., 2025a) does not have code available. Thus, we follow the evaluation pipeline from VoMP (Dagli et al., 2025) and faithfully reproduce the parts, "Material Grouping and Internal Discovery" and "MLLMs-Guided Material Identification". We reproduce these parts of their pipeline using GPT-4o (OpenAI & et al., 2024) for the MLLMs-Guided Material Identification. Furthermore, we obtained the prompts from the authors of Phys4DGen (Lin et al., 2025a) and use the same prompts.

H. Additional Details on the Simulations

We experiment with Simplicits (Modi et al., 2024), a reduced order simulator (Modi et al., 2024; Xiang et al., 2026), an accurate finite-element method (FEM) (Huang et al., 2025; 2024a) simulator, and Isaac (NVIDIA) for our simulations with generated mechanical properties. We share details on these simulations below.

H.1. Material Interpolation Scheme For Simulation

Material values in ADAVOMP are returned on a sparse voxel grid. Simulators, however, need material values are arbitrary query locations (such as mesh vertices or monte-carlo sampled cubature points). We used nearest-neighbor interpolation on the material voxel field to source material parameters for aribtrary query points in the field.

Table 16. Training hyperparameters for L, L+, and H.

Hyperparameter	L	L+	H
Parallelism			
GPUs (W)	32	32	32
ZeRO-3 shard group size (S_{shard})	32	32	32
DDP replica count ($R = W/S_{\text{shard}}$)	1	1	1
Optimization			
Optimizer	AdamW	AdamW	AdamW
AdamW (β_1, β_2)	$(0.9, 0.999)$	$(0.9, 0.999)$	$(0.9, 0.999)$
AdamW ϵ	10^{-8}	10^{-8}	10^{-8}
Weight decay	0.01	0.01	0.01
Training steps (T)	60,000	60,000	60,000
Batch size / GPU	1	1	1
Global batch size ($W\times$ batch/GPU)	32	32	32
Gradient accumulation steps	1	1	1
Gradient clipping (global norm)	1.0	1.0	1.0
EMA decay	0.9999	0.9999	0.9999
Precision (forward)	bf16	bf16	bf16
Learning-rate schedule			
Schedule	linear warmup + linear decay	linear warmup + linear decay	linear warmup + linear decay
Warmup init LR ($\eta_{\text{peak}}/T_{\text{w}}$)	10^{-7}	10^{-7}	10^{-7}
LR peak (η_{peak})	2×10^{-4}	2×10^{-4}	2×10^{-4}
Warmup steps (T_{w})	2,000	2,000	2,000
LR end (η_{end})	2×10^{-6}	2×10^{-6}	2×10^{-6}
Loss			
Structure loss $\mathcal{L}_{\text{struct}}$	3-way cross-entropy	3-way cross-entropy	3-way cross-entropy
Structure loss weight λ_{struct}	1	1	1
Material loss $\mathcal{L}_{\text{mat}}$	ℓ_2 on normalized triplets	ℓ_2 on normalized triplets	ℓ_2 on normalized triplets
Level-weight base γ (for $\omega_\ell = \gamma^\ell$)	1.4	1.4	1.4
Material loss weight λ_{mat}	75	75	75
Per-property weights $\mathbf{\Lambda}$	$(1, 1, 3)$	$(1, 1, 3)$	$(1, 1, 3)$
Sparsity and exact chunking (output-preserving)			
Max candidates / level ($\lvert\mathcal{C}_\ell\rvert_{\max}$)	36,192	36,192	36,192
Window-attention chunk cap (tokens, $N_{\text{chunk}}^{\text{attn}}$)	262,144	262,144	262,144
MatVAE decode chunk ($N_{\text{chunk}}^{\text{MatVAE}}$)	128	128	128

Table 17. Training compute by model scale, reported in A100-80GB GPU-days.

Scale	GPU-days
S	83
B	92
B+	92
L	166
L+	166
H	172

H.2. Evaluating on FEM and Simplicits Simulations

Our FEM and Simplicits simulation pipelines for evaluating predicted materials follows the same setup as introduced in the original VoMP paper (Dagli et al., 2025). Specifically, the predicted volumetric material parameters, (E, ν, ρ), are converted into simulation-ready Lamé parameters and passed to the object's constitutive model. The FEM and Simplicits solver implementation details are identical to the original VoMP paper, as well as solver hyperparameters and pre-processing steps (such as mesh construction and simplicits basis construction). Our FEM hyperparameters are listed in Table 18. All out simulations are run on an RTX A6000 - 48 GB.

H.3. Evaluating on IsaacSim

Additional simulations are conducted in NVIDIA Isaac Sim (NVIDIA), a robotics simulation platform built on NVIDIA Omniverse that leverages the PhysX physics engine (NVIDIA Corporation, 2025d) for rigid body dynamics. Tb. 19 details the key hyperparameters used in our simulation pipeline. PhysX employs a Temporal Gauss-Seidel (TGS) iterative solver (NVIDIA Corporation, 2025d) for constraint resolution. The simulation runs at 120 Hz, maintaining real-time performance. Contact handling is enhanced through Continuous Collision Detection (CCD) on the object meshes, which prevents fast-moving objects from tunneling. Material properties, including friction coefficients and restitution values, are tuned to approximate realistic object interactions observations.

Table 18. Hyperparameters for FEM simulation.

Hyperparameter	Value	Hyperparameter	Value
Time Integrator	Backward Euler	Linear Solver	pre-conditioned CG
Nonlinear Solver	Newton's w/ line search	Linear tolerance	10^{-3}
Newton max iters.	1024	Line search	
Velocity tol.	0.05 ms^{-1}	max iters	8
CCD tol.	1.0	Collision	
Transform rate tol.	0.1/s	Friction	0.5
dt	0.02	Contact Resistance	1.0
Gravity	$[0.0, -9.8, 0.0]$	$\hat{d}$	0.01

Table 19. Hyperparameters for PhysX simulation.

Hyperparameter	Value
Solver Type	PhysX TGS
Time Step (dt)	1/120 s (0.00833 s)
Render Interval	8 steps
Position Iterations	4
Velocity Iterations	1
Relaxation	0.75
Warm Start	0.4
Gravity	$[0.0, 0.0, -9.81]$ m/s^2
Contact & Collision	
Enable CCD	True
Contact Offset	0.02 m
Rest Offset	0.01 m
Max Extraction Velocity	100.0 m/s
Shape Collision Distance	0.0 m
Shape Collision Margin	0.0 m
Shape Appx	Cvx Decomp
Object Material Properties	
Static Friction	0.5
Dynamic Friction	0.4
Restitution	0.1

I. Additional Related Works

For completeness, we include other tangentially related works here. To address the trade-off between computational efficiency and representational fidelity, recent methods across various modalities have increasingly adopted adaptive, some form of multi-scale strategies different than us. In the video and language modeling, approaches use dynamic tokenization, where models adaptively compress visual tokens (Wang et al., 2025), utilize multi-scale language units (Li et al., 2025a; Barbere et al., 2024; Nawrot et al., 2023), or learn space-time tokens to reduce temporal redundancy (Liu et al., 2017; Yan et al., 2025; Ryoo et al., 2021). Similarly, for static images, many Vision Transformers handle adaptive patch sizes, either by training a single model for variable resolutions (P & Sethi, 2022; Hu et al., 2024; Zhou & Zhu, 2023; Wang et al., 2021; 2024; Beyer et al., 2023; Chen et al., 2021) or by dynamically mixing patch sizes within a single inference pass to allocate compute to complex regions (Rao et al., 2021; Fan et al., 2024; Havtorn et al., 2023; Chen et al., 2023; An et al., 2024; Ronen et al., 2023). In 3D generation, some works generate geometry in a coarse-to-fine scheme from low-resolution priors or learn features over a 3D space in a coarse-to-fine scheme (Wei et al., 2025; Ren et al., 2024a;b; Ghadai et al., 2019; Bai et al., 2024).

Beyond static property prediction, extensive research explores recovering physical attributes through dynamic interaction or observation. This includes estimating parameters from video sequences (Davis et al., 2015; Mottaghi et al., 2016; Bhat et al., 2002; Chen et al., 2025b; Liu et al., 2024a; Xue et al., 2023; Li et al., 2025b; Brubaker et al., 2009; Yildirim et al.; Li et al., 2023; 2020b; Wu et al., 2016; 2015; 2017; Xia et al., 2024; Xu et al., 2019; Feng et al., 2023; Lin et al., 2024) or through direct physical manipulation of real-world objects (Yu et al., 2024; Pai et al., 2001; Lang et al., 2003; Lloyd & Pai, 2001; Pai et al., 2008; Pai, 2000; Yao & Hauser, 2023; Pinto et al., 2016). There are works that focus on generation, several works enforce physical plausibility such as gravitational stability during the synthesis of new shapes (Lin et al., 2025b; Guo et al., 2024; Chen et al., 2024; Ni et al., 2024; Yang et al., 2024; Mezghanni et al., 2022; Chen et al., 2025a; Cao et al., 2025; Cao & Kalogerakis, 2025). However, unlike our approach, these methods focus on creating new geometry rather than augmenting existing assets with mechanical fields. Finally, alternative techniques bypass explicit property estimation entirely by directly predicting surface displacements (Zhang et al., 2024; Shi et al., 2023) or articulation parameters (Xia et al., 2025; Goyal et al., 2025; Song et al., 2025; Aygun & Mac Aodha, 2024; Werby et al., 2025; Li et al., 2020a).

Algorithm 2 Material Tree Construction via Value-Range Refinement

Require: Finest-level occupied indices $\mathbf{I}_0 \in \mathbb{Z}^{N_0 \times 3}$, materials $\mathbf{M}_0 \in \mathbb{R}^{N_0 \times 3}$, resolution $G = 2^{L_{\max}}$, tolerance $\boldsymbol{\tau} \in \mathbb{R}^3_+$

Ensure: Stored material tree $\mathcal{T} = \bigcup_\ell \{(\ell, \mathbf{i}, \mathbf{m}_{\ell,\mathbf{i}})\}$ (leaves plus internal supervision nodes)

Phase 1: bottom-up statistics (aligned by parent hashing)
$\mathbf{C}^{(0)} \leftarrow \mathbf{I}_0$
$\mathbf{m}^{(0)} \leftarrow \mathbf{M}_0$
$\mathbf{m}^{\min(0)} \leftarrow \mathbf{M}_0$
$\mathbf{m}^{\max(0)} \leftarrow \mathbf{M}_0$
for $\ell = 1, 2, \ldots, L_{\max}$ **do**
  $G_\ell \leftarrow G/2^\ell$
  Parent coords
  $\mathbf{P} \leftarrow \lfloor \mathbf{C}^{(\ell-1)}/2 \rfloor$
  $\mathbf{h} \leftarrow h_\ell(\mathbf{P})$
  Group children by parent hash
  $(\mathbf{u}, \mathbf{inv}, \mathbf{cnt}) \leftarrow \textsc{Unique}(\mathbf{h})$
  $\mathbf{C}^{(\ell)} \leftarrow \textsc{Unhash}(\mathbf{u}, G_\ell)$
  $\mathbf{m}^{(\ell)} \leftarrow \textsc{ScatterSum}(\mathbf{m}^{(\ell-1)}, \mathbf{inv})/\mathbf{cnt}$
  $\mathbf{m}^{\min(\ell)} \leftarrow \textsc{ScatterMin}(\mathbf{m}^{\min(\ell-1)}, \mathbf{inv})$
  $\mathbf{m}^{\max(\ell)} \leftarrow \textsc{ScatterMax}(\mathbf{m}^{\max(\ell-1)}, \mathbf{inv})$
  Descendant range
  $\Delta^{(\ell)} \leftarrow \mathbf{m}^{\max(\ell)} - \mathbf{m}^{\min(\ell)}$
end for
Phase 2: coarse-to-fine selection of stored nodes
$\ell_{\text{start}} \leftarrow \max\{\ell : |\mathbf{C}^{(\ell)}| > 0\}$
H[ℓ]: hashes of subdivided voxels at level ℓ
$\mathcal{T} \leftarrow \emptyset$
$\mathcal{H} \leftarrow \emptyset$
for $\ell = \ell_{\text{start}}$ **downto** 0 **do**
  if $\ell = \ell_{\text{start}}$ **then**
    $\texttt{active} \leftarrow \texttt{True}^{|\mathbf{C}^{(\ell)}|}$
  else
    $\mathbf{p} \leftarrow \lfloor \mathbf{C}^{(\ell)}/2 \rfloor$
    $\mathbf{h}_p \leftarrow h_{\ell+1}(\mathbf{p})$
    $\texttt{active} \leftarrow \textsc{IsIn}(\mathbf{h}_p, \mathcal{H}[\ell+1])$
  end if
  $\mathbf{C}_a \leftarrow \mathbf{C}^{(\ell)}[\texttt{active}]$
  $\mathbf{m}_a \leftarrow \mathbf{m}^{(\ell)}[\texttt{active}]$
  if $\ell = 0$ **then**
    Add all $(0, \mathbf{i}, \mathbf{m}_a)$ to $\mathcal{T}$
  else
    $\texttt{subdiv} \leftarrow \textsc{Any}(\Delta^{(\ell)}[\texttt{active}] > \boldsymbol{\tau})$
    Coarse leaves
    Add all $(\ell, \mathbf{i}, \mathbf{m}_a[\neg\texttt{subdiv}])$ to $\mathcal{T}$
    Internal supervision
    Add all $(\ell, \mathbf{i}, \mathbf{m}_a[\texttt{subdiv}])$ to $\mathcal{T}$
    $\mathcal{H}[\ell] \leftarrow h_\ell(\mathbf{C}_a[\texttt{subdiv}])$
  end if
end for
return $\mathcal{T}$

Algorithm 3 Conditioning Feature Tree Construction via Lazy Refinement

Require: Finest-level occupied indices $\mathbf{I}_0 \in \mathbb{Z}^{N_0 \times 3}$, grid size $G = 2^{L_{\max}}$, start level ℓ_{start}, samples per cell K, uniformity threshold $\tau_{\text{feat}} > 0$, max nodes $N_{\max}$, and voxel feature lifting as in Equation (18)

Ensure: Frontier-only feature tree $\mathcal{T} = \bigcup\{(\ell, \mathbf{i}, \mathbf{e}_{\ell,\mathbf{i}})\}$

$\mathcal{T} \leftarrow \emptyset$
$\texttt{frontier} \leftarrow \{(\ell_{\text{start}}, \mathbf{c}) : \mathbf{c} \in \textsc{Unique}(\lfloor \mathbf{I}_0/2^{\ell_{\text{start}}} \rfloor)\}$
while $|\texttt{frontier}| > 0$ **and** $|\mathcal{T}| < N_{\max}$ **do**
  Pop $(\ell, \mathbf{c})$ with maximal ℓ from $\texttt{frontier}$
  $\mathcal{S} \leftarrow \{n : \lfloor \mathbf{I}_0[n]/2^\ell \rfloor = \mathbf{c}\}$
  if $|\mathcal{S}| = 0$ **then**
    continue
  end if
  $\mathcal{S}_{\text{samp}} \leftarrow \textsc{Sample}(\mathcal{S}, \min\{K, |\mathcal{S}|\})$
  $\mathbf{P} \leftarrow (\mathbf{I}_0[\mathcal{S}_{\text{samp}}] + 0.5)/G - 0.5$
  $\mathbf{E} \leftarrow \textsc{LiftDINO}(\mathbf{P})$ (uses Equation (18))
  Row-normalize features: $\mathbf{E}_n \leftarrow \mathbf{E}/\|\mathbf{E}\|_2$
  $\mathbf{D} \leftarrow \sqrt{2 - 2(\mathbf{E}_n \mathbf{E}_n^\top)}$
  $\texttt{uniform} \leftarrow \left(\max_{a \neq b} \mathbf{D}_{a,b} \leq \tau_{\text{feat}}\right)$
  if $\texttt{uniform}$ **or** $\ell = 0$ **then**
    $\mathbf{e}_{\ell,\mathbf{c}} \leftarrow \textsc{Mean}(\mathbf{E})$
    Add $(\ell, \mathbf{c}, \mathbf{e}_{\ell,\mathbf{c}})$ to $\mathcal{T}$
  else
    for each $\mathbf{o} \in \{0, 1\}^3$ **do**
      $\mathbf{c}' \leftarrow 2\mathbf{c} + \mathbf{o}$
      if child cell $(\ell - 1, \mathbf{c}')$ contains at least one occupied voxel **then**
        Add $(\ell - 1, \mathbf{c}')$ to $\texttt{frontier}$
      end if
    end for
  end if
end while
return $\mathcal{T}$

Algorithm 4 Batched Point Query (Finest-Available)

Require: Query points $\mathbf{X} \in \Omega^{Q\times 3}$, per-level coordinate sets $\{\mathcal{I}_\ell\}$, per-level features $\{\mathbf{F}_\ell\}$, grid size $G = 2^{L_{\max}}$
Ensure: Queried features $\mathbf{R} \in \mathbb{R}^{Q\times d}$

$\mathbf{R} \leftarrow \mathbf{0}$
$\texttt{found} \leftarrow \texttt{False}^Q$
for $\ell = 0, 1, \ldots, L_{\max}$ **do**
  Finest to coarsest
  if ALL($\texttt{found}$) **then**
    break
  end if
  $G_\ell \leftarrow G/2^\ell$
  Level indices
  $\mathbf{I}_q \leftarrow \lfloor (\mathbf{X} + 0.5)/s_\ell \rfloor$
  $\mathbf{h}_q \leftarrow h_\ell(\mathbf{I}_q)$
  $\mathbf{h}_\ell \leftarrow h_\ell(\mathcal{I}_\ell)$
  $(\mathbf{h}_s, \mathbf{perm}) \leftarrow \textsc{Sort}(\mathbf{h}_\ell)$
  $\mathbf{idx} \leftarrow \textsc{SearchSorted}(\mathbf{h}_s, \mathbf{h}_q)$
  clamp $\mathbf{idx}$ to valid range
  $\texttt{match} \leftarrow (\mathbf{h}_s[\mathbf{idx}] = \mathbf{h}_q)$
  $\texttt{upd} \leftarrow \texttt{match} \wedge \neg\texttt{found}$
  $\mathbf{R}[\texttt{upd}] \leftarrow \mathbf{F}_\ell[\mathbf{perm}[\mathbf{idx}[\texttt{upd}]]]$
  $\texttt{found}[\texttt{upd}] \leftarrow \texttt{True}$
end for
if $\neg$ALL($\texttt{found}$) **then**
  Assign remaining queries by nearest-neighbor over voxel centers across all stored nodes
end if
return $\mathbf{R}$

Algorithm 5 Inference-Time Coarse-to-Fine Decoding

Require: Conditioning latents $\mathbf{H}^{\text{in}}$ from AGT; max level $L_{\max}$; initial candidate set $\mathcal{C}_{L_{\max}} = \{(L_{\max}, (0, 0, 0))\}$.
Ensure: Generated material tree $\mathcal{T}^{\mathcal{M}'}$.

Initialize $\mathcal{T}^{\mathcal{M}'} \leftarrow \emptyset$
Initialize $\mathcal{C}_{L_{\max}} \leftarrow \{(L_{\max}, (0, 0, 0))\}$
Initialize parent-to-child state for the root as $\mathbf{0}$ (as in Equation (4))
for $\ell = L_{\max}, L_{\max} - 1, \ldots, 0$ **do**
  Run AMG at level ℓ on candidates $\mathcal{C}_\ell$ (queries by Equation (4)) to obtain logits $\{\mathbf{a}_{\ell,\mathbf{i}}\}$ and latent means $\{\hat{\mathbf{z}}_{\ell,\mathbf{i}}\}$.
  $\hat{s}_{\ell,\mathbf{i}} \leftarrow \arg\max \text{softmax}(\mathbf{a}_{\ell,\mathbf{i}})$ for each candidate.
  Enforce boundary constraints: at $\ell = L_{\max}$, disallow EMPTY; at $\ell = 0$, disallow SUBDIVIDE.
  Add all non-empty candidates to $\mathcal{T}^{\mathcal{M}'}$ with their predicted latents $\hat{\mathbf{z}}_{\ell,\mathbf{i}}$ (MatVAE decoding in Section 4.2).
  if $\ell > 0$ **then**
    $\mathcal{C}_{\ell-1} \leftarrow \bigcup_{(\ell,\mathbf{i})\in\mathcal{C}_\ell:\ \hat{s}_{\ell,\mathbf{i}}=\textsc{Subdivide}} \text{Children}(\ell, \mathbf{i})$
    if $\mathcal{C}_{\ell-1} = \emptyset$ **then**
      break
    end if
  end if
end for
return $\mathcal{T}^{\mathcal{M}'}$

Algorithm 6 Window-Based Candidate Capping

Require: Candidate grid indices $\mathbf{I} \in \mathbb{Z}^{N\times 3}$ and aligned parent-to-child states $\mathbf{P} \in \mathbb{R}^{N\times d}$ (set to $\mathbf{0}$ at the root); cap $N_{\max}$; minimum window size w.
Ensure: Subsampled candidates $(\mathbf{I}', \mathbf{P}')$ and selected indices π.

if $N \le N_{\max}$ **then**
  $\pi \leftarrow \{1, \dots, N\}$
  return $(\mathbf{I}, \mathbf{P}),\ \pi$
end if
Compute spatial bounding box:
$\mathbf{i}_{\min} \leftarrow \min(\mathbf{I})$
$\mathbf{i}_{\max} \leftarrow \max(\mathbf{I})$
$\boldsymbol{\delta} \leftarrow \mathbf{i}_{\max} - \mathbf{i}_{\min} + \mathbf{1}$
Choose a centered window sized to contain $\approx N_{\max}$ points.
$r \leftarrow N_{\max}/N$
$\alpha \leftarrow r^{1/3}$
$\mathbf{w} \leftarrow \max(w,\ \lfloor \alpha\, \boldsymbol{\delta} \rfloor)$
$\mathbf{c} \leftarrow \lfloor (\mathbf{i}_{\min} + \mathbf{i}_{\max})/2 \rfloor$
$\mathbf{o} \leftarrow \mathrm{clip}(\mathbf{c} - \lfloor \mathbf{w}/2 \rfloor,\ \mathbf{i}_{\min},\ \mathbf{i}_{\max} - \mathbf{w} + \mathbf{1})$
$\mathbf{o}^{+} \leftarrow \mathbf{o} + \mathbf{w}$
Keep candidates in the window.
$\pi \leftarrow \{n :\ \mathbf{o} \le \mathbf{I}_n < \mathbf{o}^{+}\}$ (componentwise)
if $|\pi| > N_{\max}$ **then**
  Truncate π to its first $N_{\max}$ indices.
end if
if $|\pi| < N_{\max}/4$ **then**
  Fallback $\pi \leftarrow \{1, \dots, \min(N, N_{\max})\}$.
end if
$\mathbf{I}' \leftarrow \mathbf{I}[\pi]$
$\mathbf{P}' \leftarrow \mathbf{P}[\pi]$
return $(\mathbf{I}', \mathbf{P}'),\ \pi$

Algorithm 7 Teacher-Forced Candidate Expansion and Loss Evaluation During Training

Require: Conditioning latents $\mathbf{H}^{\text{in}}$ from AGT; ground-truth material tree $\mathcal{T}^{\mathcal{M}}$ with per-level voxel sets $\{\mathcal{V}_\ell^\star\}$; max level $L_{\max}$; candidate cap $|\mathcal{C}_\ell|_{\max}$; level weights ω_ℓ; loss weights $(\lambda_{\text{struct}}, \lambda_{\text{mat}})$ and $\boldsymbol{\Lambda}$.
Ensure: Loss $\mathcal{L}$ as in Equation (6).

Initialize frontier
$\mathcal{C}_{L_{\max}} \leftarrow \{(L_{\max}, (0,0,0))\}$
`sum_CE` $\leftarrow 0$
`sum_MSE` $\leftarrow 0$
`cnt_C` $\leftarrow 0$
`cnt_P` $\leftarrow 0$
for $\ell = L_{\max}, L_{\max}-1, \dots, 0$ **do**
  Candidate capping If $|\mathcal{C}_\ell| > |\mathcal{C}_\ell|_{\max}$, subsample candidates with a contiguous spatial window (Algorithm 6).
  Run AMG at level ℓ on candidates $\mathcal{C}_\ell$ conditioned on $\mathbf{H}^{\text{in}}$ to obtain logits $\{\mathbf{a}_{\ell,\mathbf{i}}\}$ and latents $\{\mathbf{z}_{\ell,\mathbf{i}}\}$.
  for each $(\ell, \mathbf{i}) \in \mathcal{C}_\ell$ **do**
    Compute ground-truth structure label $s^\star_{\ell,\mathbf{i}}$ by Equation (7).
    `sum_CE` $+= \omega_\ell\big(-\log \mathrm{softmax}(\mathbf{a}_{\ell,\mathbf{i}})(s^\star_{\ell,\mathbf{i}})\big)$
    `cnt_C` $+= 1$
    if $s^\star_{\ell,\mathbf{i}} \neq$ EMPTY **then**
      Decode $\mathbf{z}_{\ell,\mathbf{i}}$ with frozen MatVAE to obtain $\hat{\mathbf{m}}_{\ell,\mathbf{i}}$ and compare to target $\mathbf{m}^\star_{\ell,\mathbf{i}}$.
      `sum_MSE` $+= \omega_\ell \left\|\hat{\mathbf{m}}_{\ell,\mathbf{i}} - \mathbf{m}^\star_{\ell,\mathbf{i}}\right\|^2_{\boldsymbol{\Lambda}}$
      `cnt_P` $+= 1$
    end if
  end for
  if $\ell > 0$ **then**
    Teacher-forced expansion
    $\mathcal{C}_{\ell-1} \leftarrow \bigcup_{(\ell,\mathbf{i})\in\mathcal{C}_\ell:\ s^\star_{\ell,\mathbf{i}}=\text{SUBDIVIDE}} \mathrm{Children}(\ell, \mathbf{i})$ (see Equation (5))
  end if
end for
$\mathcal{L}_{\text{struct}} \leftarrow$ `sum_CE`/`cnt_C`
$\mathcal{L}_{\text{mat}} \leftarrow$ `sum_MSE`/`cnt_P`
return $\mathcal{L} \leftarrow \lambda_{\text{struct}}\mathcal{L}_{\text{struct}} + \lambda_{\text{mat}}\mathcal{L}_{\text{mat}}$

Figure 13. **Inferred Mechanical Property Fields.** We show additional mechanical property fields and slice planes through mechanical property fields estimated by ADAVOMP.

Figure 14. **Inferred Mechanical Property Fields.** We show additional mechanical property fields and slice planes through mechanical property fields estimated by ADAVOMP.

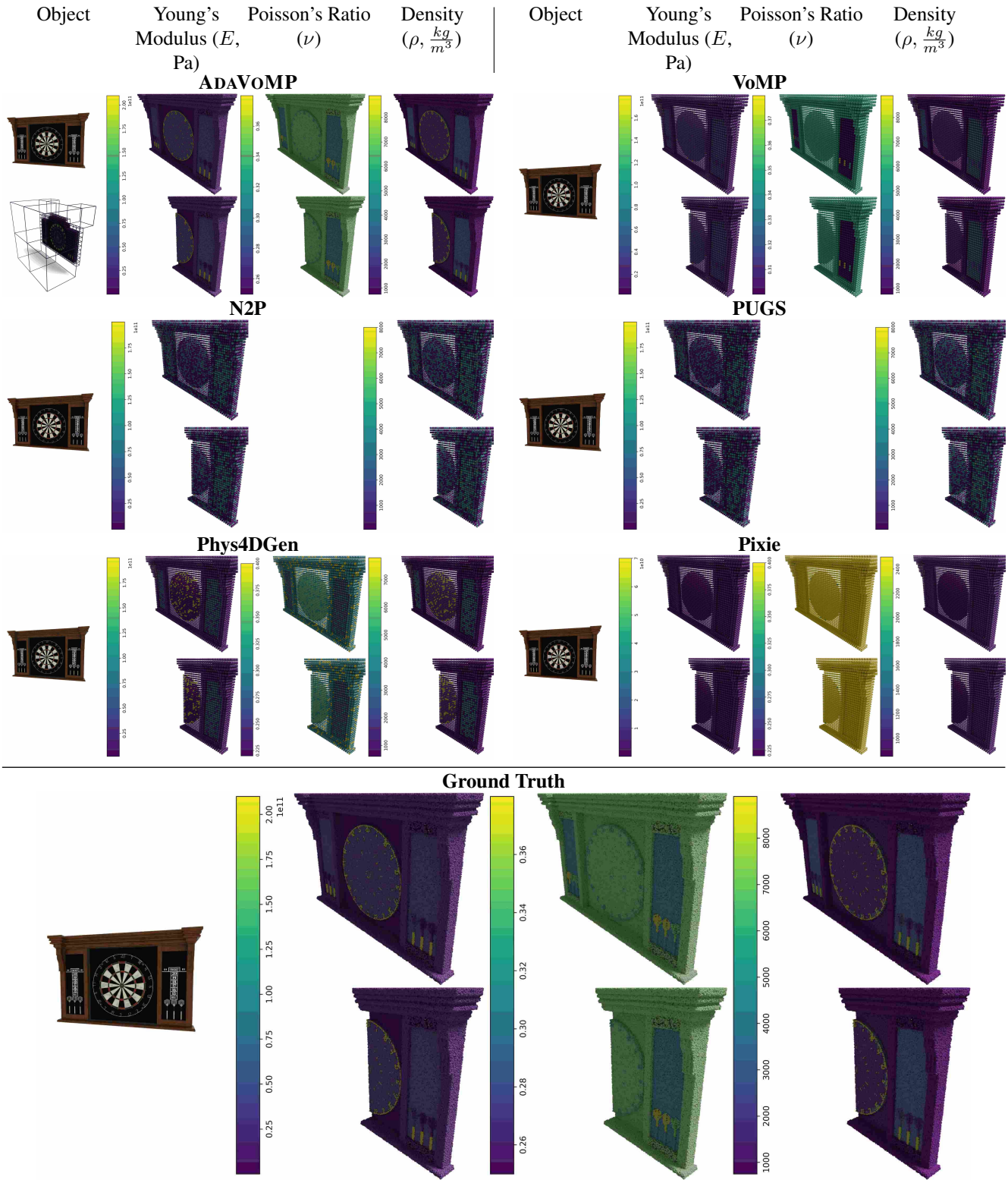


Figure 15. **Dartboard Comparison.** Mechanical property field comparisons across different methods.

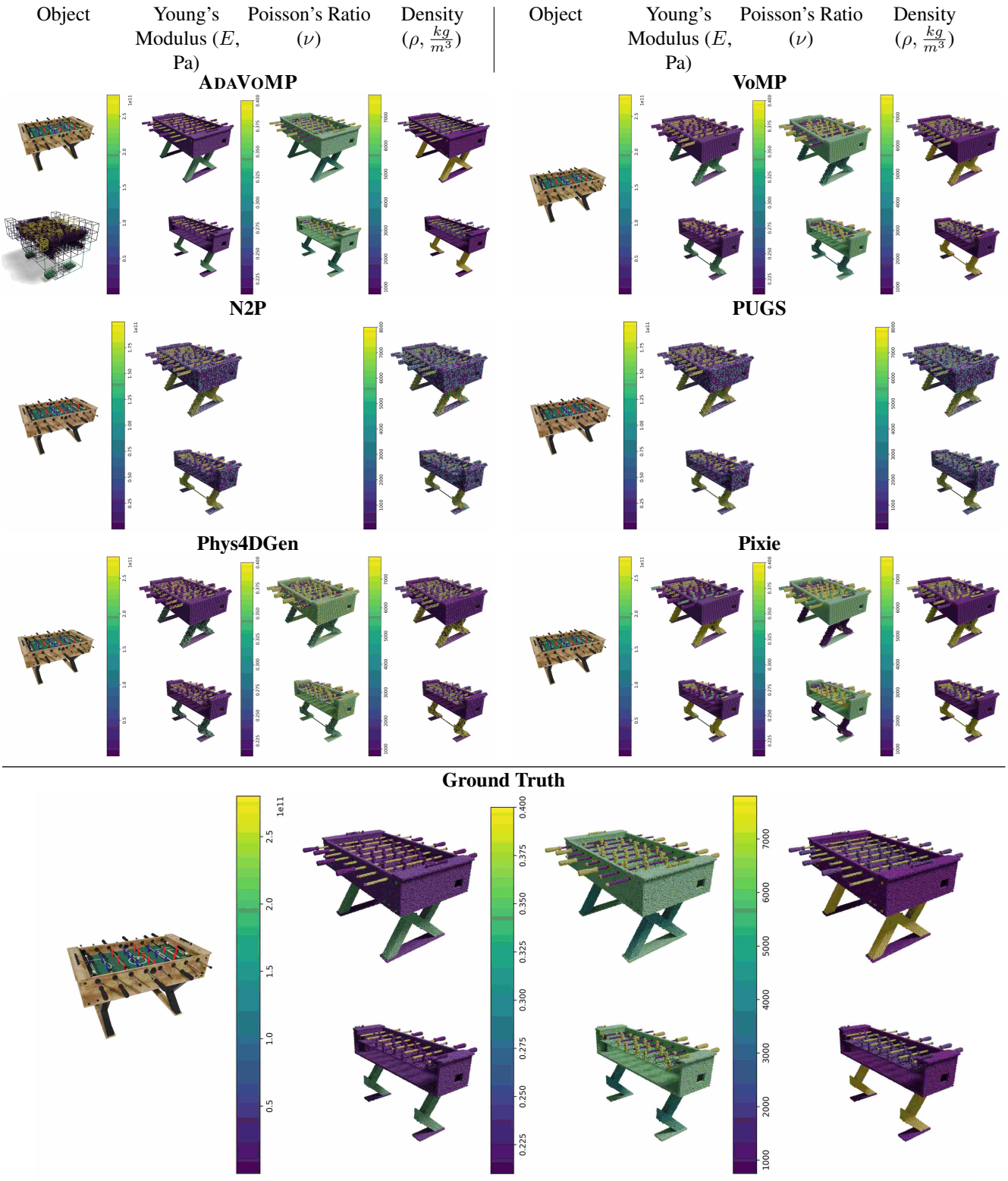


Figure 16. **Foosball Comparison.** Mechanical property field comparisons across different methods.

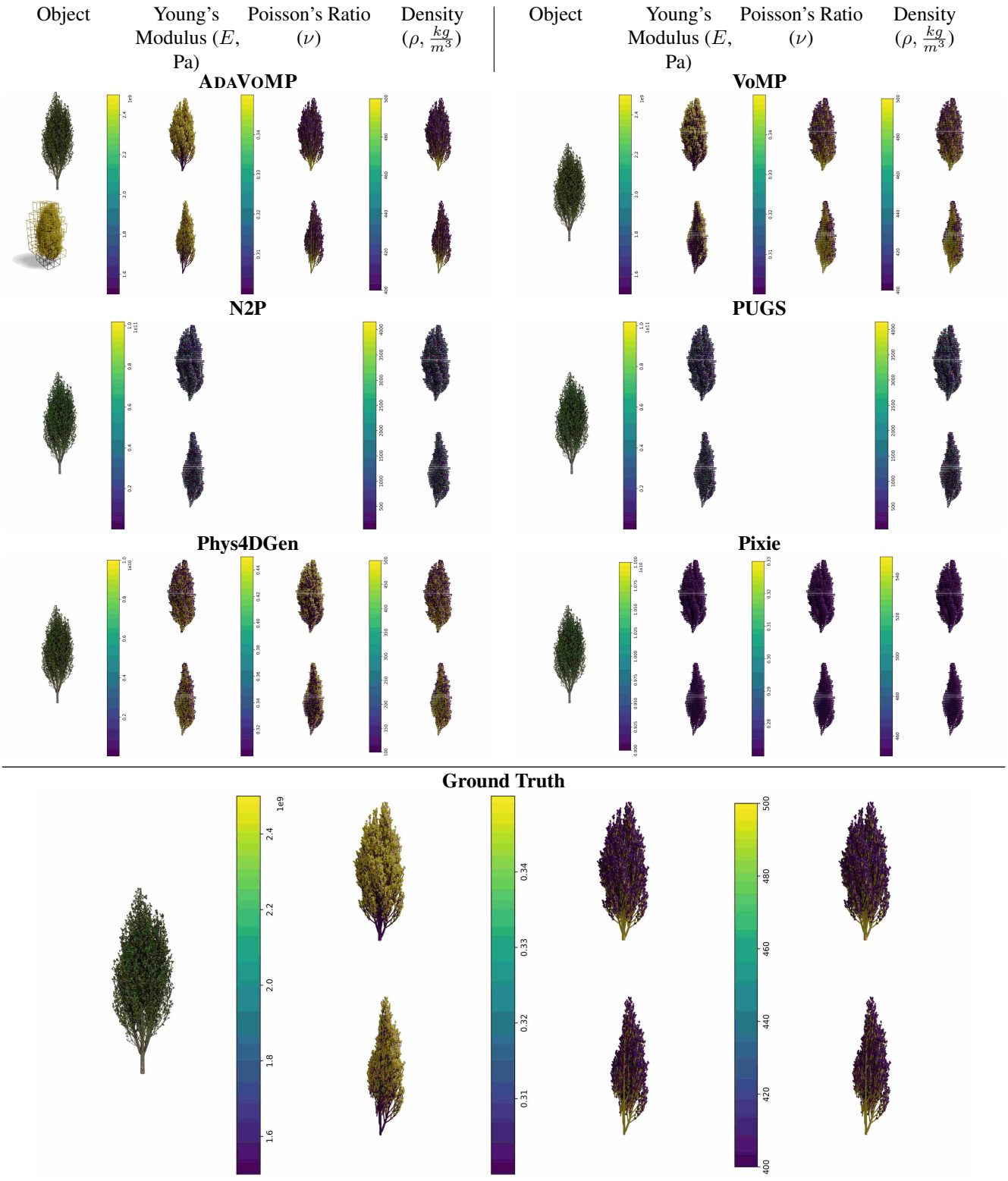


Figure 17. **Lombardy Poplar Comparison.** Mechanical property field comparisons across different methods.

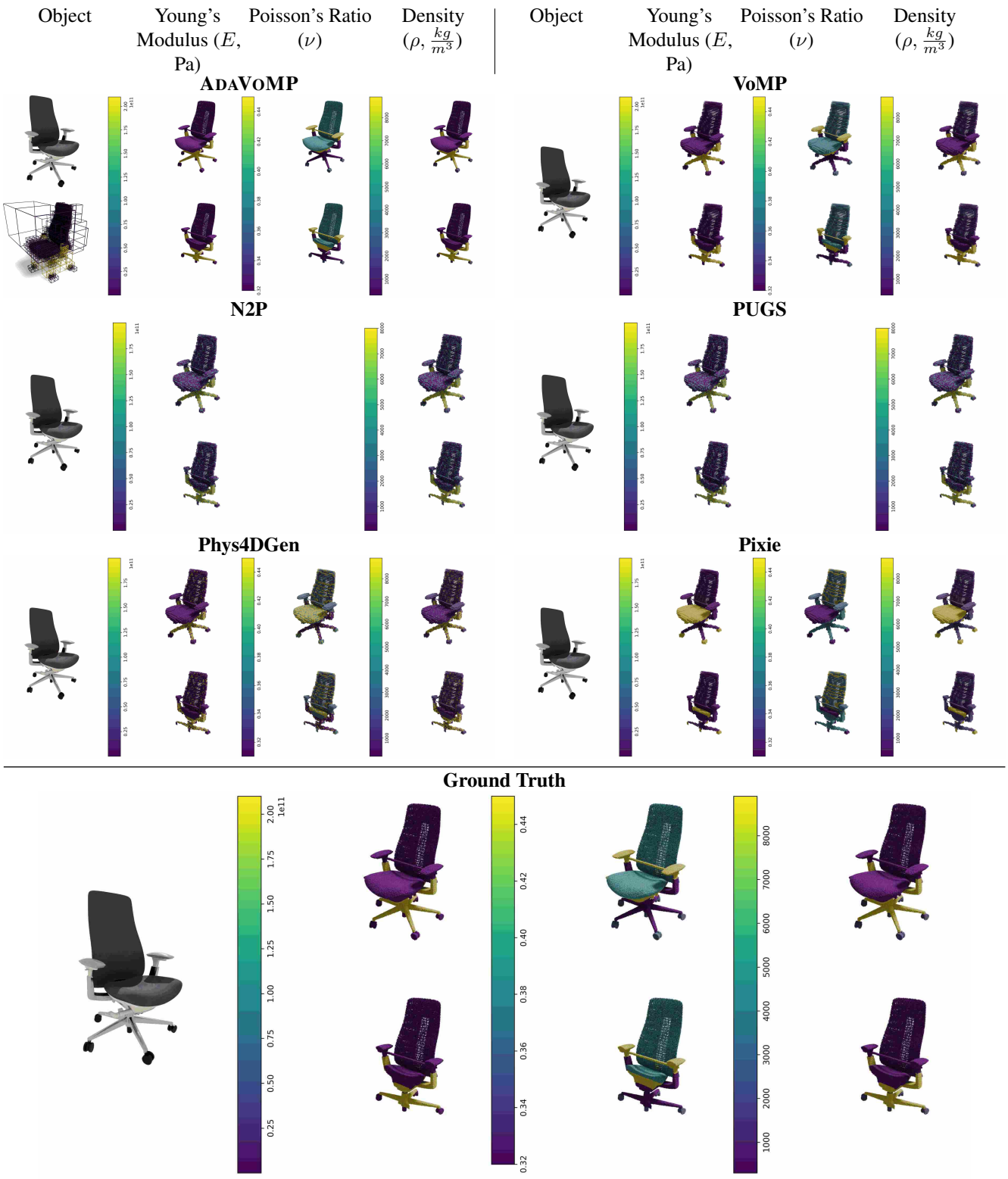
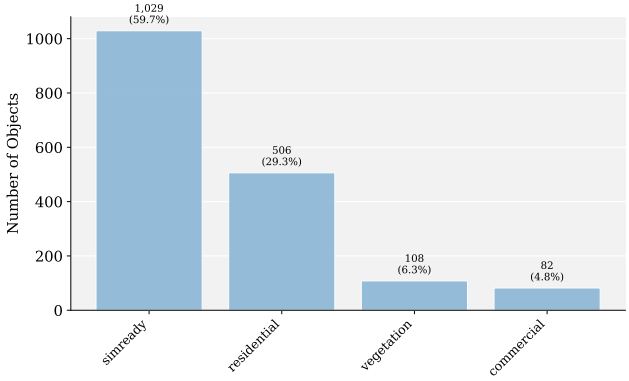
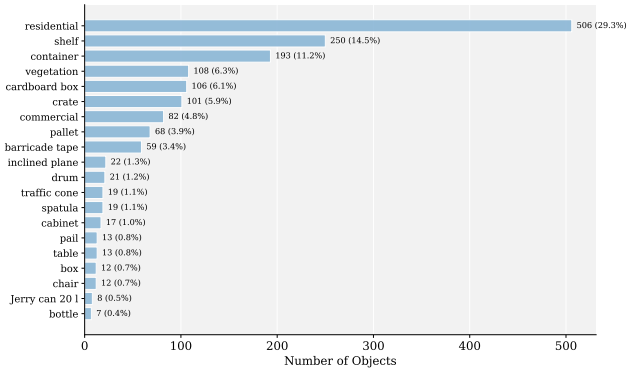


Figure 18. **Phineas Comparison.** Mechanical property field comparisons across different methods.

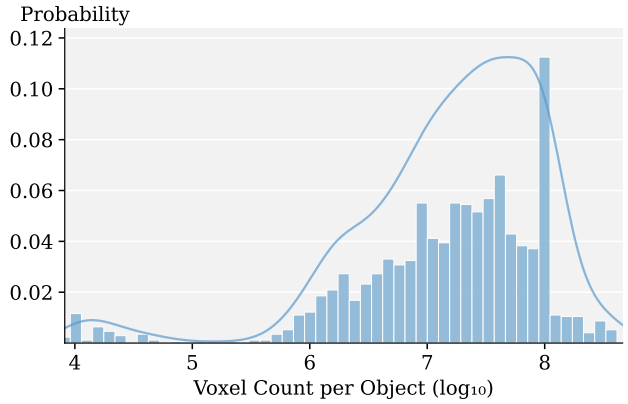
Figure 19. **Shield Controller Comparison.** Mechanical property field comparisons across different methods.



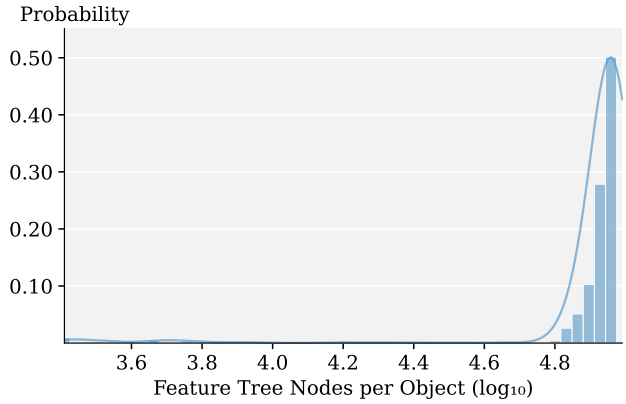
(a) Source-collection distribution.



(b) Semantic class distribution (top classes).

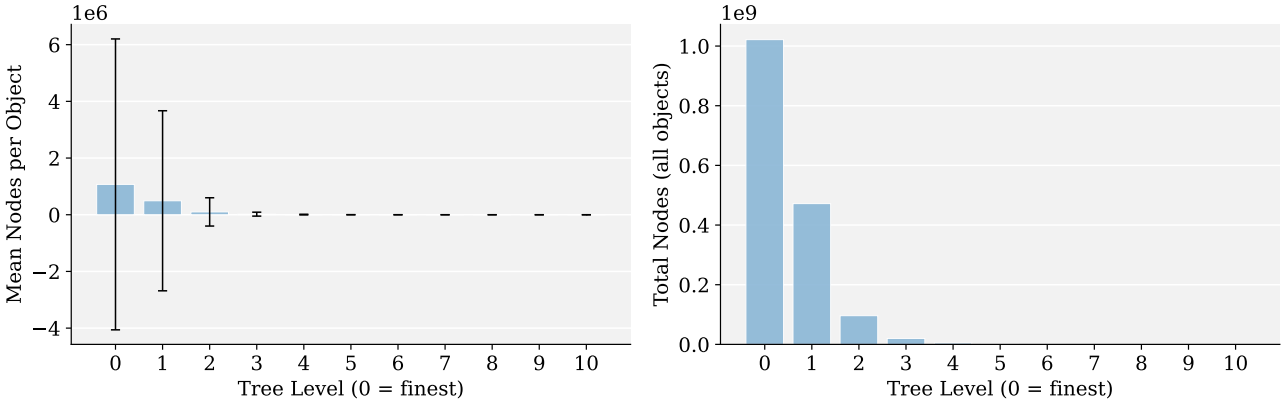


(c) Occupied voxel count per object.

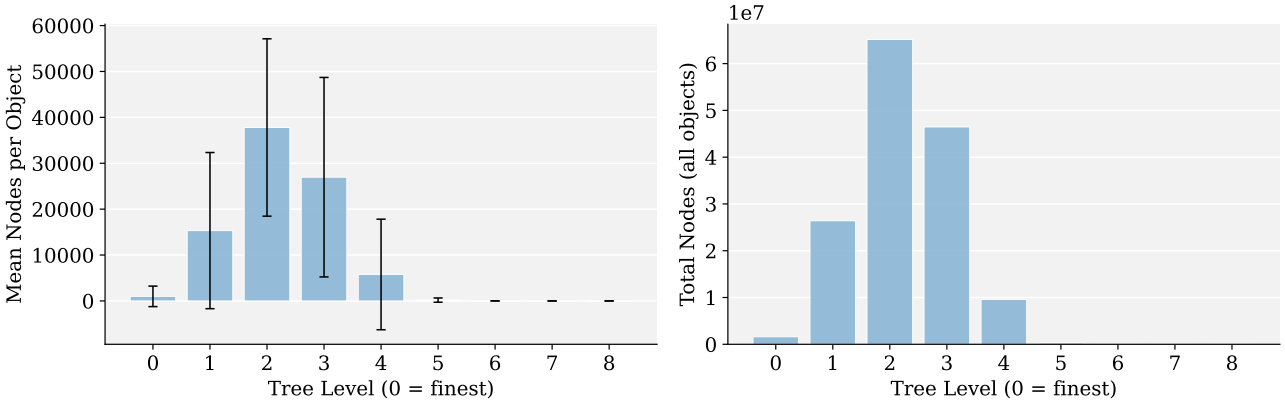


(d) Feature-tree nodes per object.

Figure 20. Dataset statistics for GVT computed after preprocessing at $G = 1024$.



(a) Material tree.



(b) Feature tree.

Figure 21. Distribution of SAV tree nodes across levels in GVT. Each plot reports (left) the mean nodes per object at each level and (right) the total nodes aggregated over all objects.

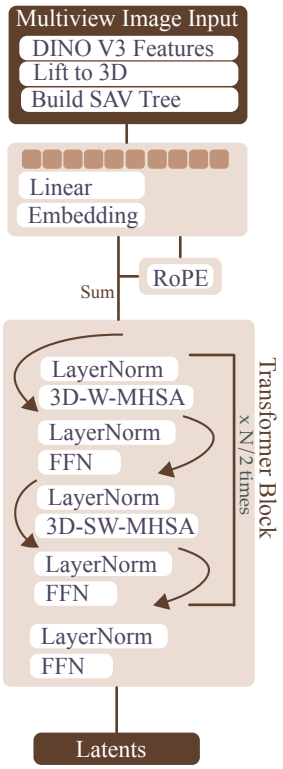


Figure 22. Encoder Network

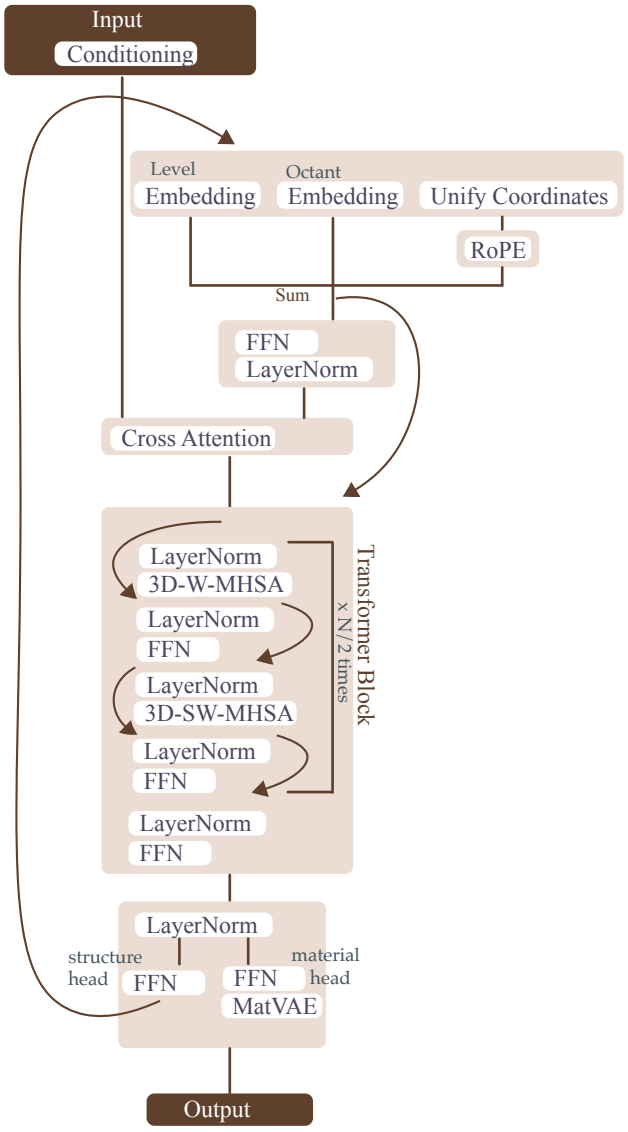


Figure 23. Decoder Network