UniPixie: Unified and Probabilistic 3D Physics Learning via Flow Matching

Supplementary Material

Qilin Huang*†[1,2], Quynh Anh Huynh*[1], Long Le*[1], Chen Wang[1], Chuhao Chen[1],
Ryan Lucas[3], Eric Eaton[1], Lingjie Liu[1]
∗ Denotes equal contribution † Work done during an internship at UPenn
[1]University of Pennsylvania [2]Southern University of Science and Technology [3]MIT
huangqilin2022@mail.sustech.edu.cn, ryanlu@mit.edu
{qanh308, vlongle, chenw30, chuhaoc, eeaton, lingjie.liu}@seas.upenn.edu

This supplementary material is organised as follows:

- **A PIXIEMULTIVERSE Details** — VLM annotation pipeline, including the explicit physical constraints, quality audit, systematic bias analysis, and cross-solver parameter mapping.
- **B Model Architecture and Training** — Provides detailed architecture specifications, training objectives, hyperparameters, inference procedure, and joint vs. separate training ablation.
- **C Baseline Implementation Details** — Details the zero-shot VLM adaptations and the fully automated VLM-feedback generative baseline.
- **D Additional Ablation Studies** — Presents a detailed continuous per-α comparison against PIXIE, and a detailed FMT vs. U-Net decoder ablation.

A. Dataset Details

This section provides a comprehensive description of PIXIEMULTIVERSE, covering the full annotation pipeline, cross-solver parameter generation strategy, and a rigorous reliability analysis of the VLM-based labeling process. Our work re-annotates the 3D assets of PIXIEVERSE [5] under a fundamentally new paradigm: instead of predicting a single point estimate, we label *plausible property ranges* $[\boldsymbol{y}_{\min}, \boldsymbol{y}_{\max}]$ and generate consistent ground-truth parameters for *multiple physics solvers*. Our MPM simulations use the same PhysGaussian solver configuration and external force scenarios (wind test for vegetation, free-fall for rigid/elastic objects) as PIXIE [5].

A.1. Annotation Pipeline for PIXIEMULTIVERSE

Our semi-automatic pipeline extends PIXIE [5] with two key novelties: (1) annotating a continuous range for each physical property rather than a single value, and (2) generating dynamically consistent ground-truth parameters for LBS and Spring-Mass solvers by fitting them to MPM-driven reference simulations.

A.1.1. MPM Range Annotation

We employ a two-stage Actor-Critic pipeline that combines strong VLM priors with rigorous human verification.

Stage 1 — Actor VLM Range Proposal. **GPT-4o** serves as the Actor. Given multi-view renders and a short semantic label, GPT-4o is prompted (Figure S1) to

1. decompose the object into *functional* parts that differ in expected physical behavior (*e.g.*, *trunk*, *leaves*, *pot*);
2. propose a plausible interval $[v_{\min}, v_{\max}]$ for E, ρ, and ν, enforcing explicit MPM simulator rules (*e.g.*, providing realistic upper bounds for extreme rigidity to ensure simulation stability);
3. emit lists of CLIP-friendly segmentation synonyms for each part; and
4. write Pythonic assert statements to guarantee cross-part physical consistency.

The full system prompt and an in-context JSON example are shown in Figure S1.

Stage 2 — Critic VLM Segmentation Validation. **Gemini-2.5-Flash** serves as the Critic. It receives the rendered 3D segmentation maps produced by each candidate CLIP-query set and evaluates geometric coherence. Each voxel is assigned to the part whose CLIP query yields the highest cosine similarity (*argmax* assignment), mathematically precluding semantic redundancies (*e.g.*, *leaf* vs. *leaves* appearing as distinct parts). The Critic selects the query set with the highest coherence score; proposals with a coherence score below a threshold are flagged for human review.

Stage 3 — Simulation-Driven Refinement Loop. The candidate parameters are directly fed into the MPM solver to render dynamic videos at both the soft ($\boldsymbol{y}_{\min}$) and stiff

GPT-4o System Prompt: 3D Physics Range Annotator

SYSTEM: You are an expert Physics Simulation Engineer. Given multi-view images of a 3D object, output a physically consistent **range** of material properties — never a single scalar.

INPUTS: Multi-view RGB images + a short semantic category label.
Use *both* geometric cues (part thickness, drooping) and semantic cues (material appearance, color) to ground your predictions.

REQUIRED JSON OUTPUT FIELDS:

- `all_queries` — list of CLIP-friendly text-query lists, one sub-list per part
- `material_dict` — per-part entries containing E (Pa), ρ (kg/m^3), ν, `material_id`, and nested `reasoning`
- `constraints` — Python assert statements that must hold for *any* value sampled within the ranges

RANGE DESIGN PRINCIPLES:

- Ranges must be non-empty and create *visually distinct* soft vs. stiff boundary simulations.
- Never collapse to a single value.
- MPM stability caps: explicitly bound maximum based on material limits to prevent simulation explosion.
- Semantically impossible combinations must be excluded via constraints (*e.g.*, pot stiffer and denser than foliage).

IN-CONTEXT EXAMPLE (Bonsai tree with unglazed ceramic pot):

```
{
  "all_queries": [
    ["ceramic pot", "clay planter", "unglazed base"],
    ["woody trunk", "thick bark", "lignified stem"],
    ["green foliage", "leaf canopy", "bonsai leaves"]
  ],
  "material_dict": {
    "pot": {
      "material_id": 6,
      "density": [1800, 2400], "E": [1e7, 1e9], "nu": [0.20, 0.30],
      "reasoning": {"visual_evidence": "Thick unglazed ceramic base.",
                    "category_choice": "STATIONARY / RIGID",
                    "range_justification": "Dense and stiff."}
    },
    "trunk": {
      "material_id": 0,
      "density": [600, 900], "E": [5e4, 1e7], "nu": [0.28, 0.35],
      "reasoning": {"visual_evidence": "Rough bark, cylindrical stem.",
                    "category_choice": "ORGANIC / PLANTS",
                    "range_justification": "Firm structural wood; higher E than foliage."}
    },
    "leaves": {
      "material_id": 0,
      "density": [70, 300], "E": [7e3, 5e4], "nu": [0.35, 0.45],
      "reasoning": {"visual_evidence": "Thin, light green leaf canopy.",
                    "category_choice": "ORGANIC / PLANTS",
                    "range_justification": "Mass-scaled (low density) to prevent gravity collapse."}
    }
  },
  "constraints": "assert mat['pot']['E'][0] > mat['trunk']['E'][1];\n              assert mat['trunk']['E
      ']][0] > mat['leaves']['E'][1]"
}
```

Figure S1. **Full GPT-4o Prompt for Physical Property Range Annotation.** The Actor VLM receives detailed system instructions, visual-to-physical reasoning guidelines, and an in-context JSON example to generate per-part material ranges with enforced cross-part constraints.

($\boldsymbol{y}_{\max}$) boundaries. If an initial parameter range is numerically unstable (*e.g.*, E exceeds CFL bounds, triggering a crash) or if the soft and stiff bounds are visually indistinguishable, this generates a failure log. These engine-level error logs, alongside lightweight human visual flags (*e.g.*, "lower bound is too stiff"), are automatically fed back into a Critic VLM acting as a Feedback Agent. The agent analyzes the failure modes to automatically narrow, shift, or decouple the parametric bounds, iteratively refining the interval until a stable and visually diverse dynamic range is achieved (alternatively, human annotators can manually refine the boundary values if necessary).

Quality Audit and Bias Analysis. To validate pipeline reliability, we conducted a post-hoc audit of 100 randomly sampled objects spanning all 10 semantic categories. Two quality criteria were evaluated jointly: **Physical Plausibility** (no simulation artifacts or explosions at either boundary) and **Visual Diversity** (the soft and stiff simulations must produce perceptibly distinct dynamic behaviors).

The audit revealed three outcome categories:

Bias Source	Failure Mode	Mitigation Mechanism
Category Prior	Defaults to the modal material state for the semantic category regardless of per-instance visual cues (*e.g.*, assigns all toys plastic properties independent of surface texture or geometry).	Human verification step rejects proposals whose boundary simulations are visually indistinguishable; annotators must observe perceptibly different soft/stiff dynamics to accept.
Structural Hierarchy	Ignores inter-part physical relationships, producing similar stiffness values across structurally distinct parts (*e.g.*, $E_{\text{leaf}} \approx E_{\text{trunk}}$).	Pythonic constraint assertions are evaluated at proposal time and automatically reject any output that violates $E_{\text{trunk}}^{\min} > E_{\text{leaf}}^{\max}$ or analogous structural inequalities.
Ambiguity Collapse	Outputs an overly narrow interval or degenerates to a single scalar value, failing to capture the object's physical ambiguity.	Prompt design mandates a wide $[v_{\min}, v_{\max}]$ output format to ensure diversity; proposals that fail to produce visually distinct boundaries are automatically flagged for VLM/human refinement.

Table S1. **VLM Annotation Bias Analysis.** Three systematic bias patterns in GPT-4o material range proposals, their characteristic failure modes, and the mitigation mechanisms employed in our pipeline.

- **52.8% First-Pass Acceptance.** Proposals accepted without any modification, indicating that GPT-4o's physical priors align well with ground-truth material behaviour for the majority of objects.
- **38.3% Refinement.** Proposals requiring $\geq$1 round of human feedback. The most common triggers were insufficient range diversity (*e.g.*, a toy annotated with a narrow E interval producing visually indistinguishable endpoints) or borderline numerical instability near MPM simulation bounds.
- **8.9% Rejection.** Proposals discarded entirely due to physical logic violations (*e.g.*, $E_{\text{leaf}} > E_{\text{trunk}}$) or irrecoverable simulation collapse at one or both boundary states.

These statistics confirm that while GPT-4o provides a strong initialization, human-in-the-loop verification is essential for producing a reliable training distribution. Table S1 systematically documents the three categories of systematic bias we identified in GPT-4o's proposals and the pipeline mechanism that mitigates each.

A.1.2. Cross-Solver Parameter Generation

To train our unified architecture, we generate ground-truth parameters for the LBS and Spring-Mass solvers that are *dynamically consistent* with our primary MPM annotations. We adopt an MPM-centric strategy: solver-specific parameters are fitted to match the dynamics of reference MPM simulations at the two boundary states $\alpha \in \{0, 1\}$.

Linear Blend Skinning (LBS). We use the test-time optimisation framework from Vid2Sim [1] to fit LBS parameters to 30-frame MPM videos rendered at the soft and stiff endpoints.

- **Deformation Model.** We observe that the underlying skinning weights and control handles are nearly identical across the two stiffness levels. We therefore simplify ground-truth generation by fitting a *single* LBS deformation model to the softest MPM video ($\boldsymbol{y}_{\min}$) and treating it as the object-specific deformation basis.
- **Material Properties.** Young's Modulus E and Poisson's Ratio ν are fitted separately to each boundary video and verified to ensure dynamical consistency.
- **Topology-Property Independence.** This design ensures that θ_{LBS} is entirely α-independent: the deformation topology is fixed per object, and the full soft-to-stiff continuum is realised exclusively by modulating (E, ν) via LERP interpolation.

Spring-Mass System. We aim to learn *intrinsic* stiffness properties decoupled from extrinsic simulation parameters (damping, initial velocity).

- **Base Model.** A Spring-Gaus [12] model is fitted to the softest MPM video (30 frames, 16 camera views) to determine anchor positions and spring topology. Extrinsic parameters (global damping, initial velocity) are fixed to dataset-wide constants.
- **Continuous Softness Range.** Rather than re-fitting the entire model, a plausible continuous range is defined for the *softness vector* $\boldsymbol{\eta}$, which modulates global stiffness while keeping the spring topology fixed. This decouples intrinsic material properties from environment-specific simulation conditions.

B. Model Architecture and Training Details

This section provides a complete specification of the UNIPIXIE architecture, training procedure, and inference algorithm to support full reproducibility.

B.1. Detailed Model Architectures

UNIPIXIE consists of one shared Grid Encoder and three specialised decoder heads. Table S2 summarises all module configurations.

Table S2. **Network Configurations.** Architecture hyperparameters for the shared Grid Encoder and the three specialised decoder heads. FMT = Flow Matching Transformer. AdaLN-Zero initialises final gating weights to zero for stable training [7].

Module	# Layers	Dim (C)	# Heads	Block Architecture	Special Mechanisms
Grid Encoder	6	512	8	Cross-Attn + Self-Attn $\times 2$	Fourier PE (16 freq.)
MPM Decoder (FMT)	6	512	8	Cross-Attn + SwiGLU MLP	AdaLN-Zero, QK-Norm
LBS — HyperNetwork	4	512	—	MLP	—
LBS — Material FMT	6	512	8	Cross-Attn + SwiGLU MLP	AdaLN-Zero, QK-Norm
Spring-Mass Decoder	4	512	8	FMT (vector-mode)	AdaLN-Zero, QK-Norm

B.1.1. Grid Encoder

The Grid Encoder $\mathcal{E}$ distills the high-dimensional voxelised CLIP [8] features $\mathcal{G}_{\text{feat}} \in \mathbb{R}^{64^3 \times 768}$ into a compact set of L=64 solver-agnostic latent tokens. The architecture is inspired by Perceiver-IO [4].

3D Convolutional Stem. Two blocks of Conv3d ($4\times4\times4$, stride 2) progressively downsample the input from resolution 64^3 to 16^3 while projecting channels to the latent dimension C=512. This spatial compression manages the quadratic cost of subsequent attention and encourages the extraction of higher-level geometric structure.

Latent Tokenizer. A set of L=64 learnable query embeddings is updated by 6 cascaded Transformer blocks. Each block performs: (1) cross-attention from the latent queries *to* the flattened 16^3 convolutional features, and (2) two consecutive self-attention layers that refine the latent representation. Fourier Positional Encodings (16 frequencies per axis) are added to the flattened grid features before the first cross-attention to provide 3D spatial awareness. The output is the solver-agnostic latent $\boldsymbol{z}_{\text{latent}} \in \mathbb{R}^{64\times512}$.

B.1.2. Flow Matching Transformer (FMT) Decoder

The FMT decoder is the generative backbone shared (in architecture but not in weights) by the MPM and LBS-Material heads.

Unified Conditioning Scheme. Instead of conditioning the flow network exclusively on the integration timestep t—a standard practice in diffusion literature—we construct a fused control vector $\boldsymbol{c}$ tailored for high-fidelity, topology-aware physical simulation. As illustrated in Figure S2, $\boldsymbol{c}$ is formed by fusing three independent embeddings:

- **Timestep** $t \in [0,1]$: Embedded via sinusoidal Fourier position features (16 logarithmic frequencies $\in \{2^0\pi, \ldots, 2^8\pi\}$) followed by a two-layer SiLU MLP to map the scalar time into a C-dimensional state.
- **Control parameter** $\alpha \in [0,1]$: Unlike time, this continuous physical state scalar is mapped via a dedicated lightweight adapter (Linear→SiLU→Linear) directly into the latent space $\mathbb{R}^C$. This preserves the monotonicity required for continuous soft-to-stiff physics transitions.

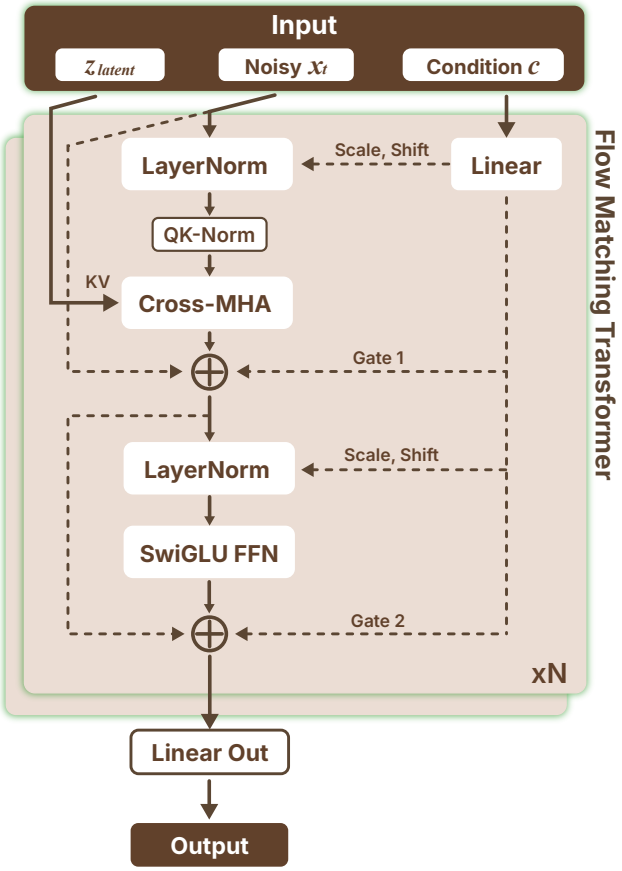

Figure S2. **Flow Matching Transformer (FMT) Architecture.** The generative backbone receives three parallel inputs: the noisy continuous state $\boldsymbol{x}_t$, localized geometry tokens $\boldsymbol{z}_{\text{latent}}$, and a unified condition vector $\boldsymbol{c}$ (fusing timestep t, physics control α, and global shape). Within the FMT Block, $\boldsymbol{z}_{\text{latent}}$ provides Keys and Values (KV) for Cross-Attention, structurally anchoring the generated flow. Simultaneously, $\boldsymbol{c}$ modulates the LayerNorms and residual connections via AdaLN-Zero to robustly steer the soft-to-stiff dynamics.

- **Global Shape Context**: We apply Global Average Pooling (GAP) across the sequence dimension of the encoder latent tokens ($\bar{\boldsymbol{z}} = \frac{1}{L}\sum_i \boldsymbol{z}_{\text{latent},i}$). This extracts a C-dimensional structure-aware semantic summary that is invariant to lo-

calized coordinate permutations.

These three embeddings are concatenated ($\mathbb{R}^{3C}$) and jointly fused through a two-layer MLP (with SiLU activation) into a single, dense condition vector $\boldsymbol{c} \in \mathbb{R}^C$. This guarantees that every downstream modulation simultaneously commands flow progression, user softness, and overarching object topology.

Transformer Block Design. Before entering the transformer blocks, the local noisy state $\boldsymbol{x}_t$ is injected with **3D Fourier Positional Encodings** to strictly preserve spatial coordinates. Each of the N=6 decoder blocks then contains two sub-layers:

- **Cross-Attention.** The noisy latent state $\boldsymbol{x}_t$ serves as queries; the full sequence of static latent tokens $\boldsymbol{z}_{\text{latent}}$ serves as keys and values. This fundamentally deviates from dense self-attention, expressly anchoring the generative state updates to the global geometry structure.
- **Feed-Forward Network.** A two-layer MLP with **SwiGLU** activation, which provides improved expressivity over standard ReLU without increasing parameter count proportionally.

Both sub-layers are modulated by **Adaptive Layer Normalisation (AdaLN)**, which regresses a sequence-wide scale γ, shift β, and gate α_{gate} directly from $\boldsymbol{c}$. We use **AdaLN-Zero** initialisation: the final linear projection for α_{gate} is initialised to zero, making each block an identity map at the start of training [7]. This significantly stabilises early training. **QK-Normalization** (RMSNorm on queries and keys before the attention softmax) prevents attention collapse in deeper networks.

B.1.3. Specialised Decoder Heads

MPM Decoder. The MPM FMT operates on the full voxel grid. At each denoising step it predicts a velocity field over K foreground voxels, each with 3 continuous channels (log-E, log-ρ, ν) and 8 categorical logits (material class). The discrete material class is predicted via a separate linear head applied to the final FMT output; all continuous properties are generated via the flow-matching ODE.

LBS Decoder (Dual-Branch). Following Vid2Sim [1], we decouple the deformation model from material properties:

1. **HyperNetwork (deformation model).** A 4-layer MLP takes the global shape embedding $\bar{\boldsymbol{z}}$ and regresses the flattened parameters $\theta_{\text{LBS}} \in \mathbb{R}^{650}$ of the skinning MLP. *Importantly, θ_{LBS} is object-specific but α-independent*: the deformation topology is fixed per object, and the soft-to-stiff continuum is realised entirely by the material FMT head. The HyperNetwork receives no α input.
2. **Material FMT head.** An FMT decoder with the same architecture as the MPM decoder predicts the spatially-varying fields (E, ν) conditioned on $(\boldsymbol{z}_{\text{latent}}, \alpha)$. These vary continuously with α; the deformation basis θ_{LBS} does not.

Spring-Mass Decoder. We build on the Spring-Gaus [12] representation. A *vector-mode* FMT decoder (same block design but operating on a single global vector rather than a spatial grid) predicts $\boldsymbol{m}_{\text{spring}} \in \mathbb{R}^{2049}$, which concatenates the per-anchor stiffness vector $\boldsymbol{k} \in \mathbb{R}^{N_a}$ (N_a=2048) with a global softness scalar $\eta \in \mathbb{R}$.

B.2. Training Objectives and Hyperparameters

The entire framework is trained end-to-end with a weighted sum of task-specific losses and one regularisation term:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{MPM}} + \lambda_{\text{LBS}}\,\mathcal{L}_{\text{LBS}} + \lambda_{\text{SM}}^{(k)}\,\mathcal{L}_{\text{SM}}^{(k)} + \lambda_{\text{SM}}^{(\eta)}\,\mathcal{L}_{\text{SM}}^{(\eta)} + \lambda_{\text{KL}}\,\mathcal{L}_{\text{KL}} \tag{S1}$$

where $\mathcal{L}_{\text{SM}}^{(k)}$ and $\mathcal{L}_{\text{SM}}^{(\eta)}$ denote the CFM losses for the per-anchor stiffness vector $\boldsymbol{k}$ and the global softness scalar η, respectively.

Conditional Flow Matching (CFM) [6] Loss. Applied to the MPM decoder, LBS-Material FMT head, and Spring-Mass decoder. For each training sample, we sample: a control value $\alpha \sim \mathcal{U}(0,1)$, a target $\boldsymbol{x}_1 = \text{LERP}(\boldsymbol{y}_{\min}, \boldsymbol{y}_{\max}; \alpha)$, noise $\boldsymbol{x}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and a flow time t from a logit-Normal distribution [2]. The model predicts the velocity $\boldsymbol{v}_\theta$ optimised via:

$$\mathcal{L}_{\text{CFM}} = \mathbb{E}_{t,\boldsymbol{x}_0,\boldsymbol{x}_1,\boldsymbol{c}} \left\| \boldsymbol{v}_\theta(\boldsymbol{x}_t, t, \boldsymbol{c}) - (\boldsymbol{x}_1 - \boldsymbol{x}_0) \right\|_2^2 \tag{S2}$$

where $\boldsymbol{x}_t = (1-t)\boldsymbol{x}_0 + t\boldsymbol{x}_1$.

LBS Regression Loss. The HyperNetwork branch is trained with an MSE loss between the predicted $\hat{\theta}_{\text{LBS}}$ and the interpolated ground-truth parameters. Because θ_{LBS} is α-independent, no α conditioning is used here.

KL Consistency Regularisation. A KL divergence term ($\lambda_{\text{KL}} = 10^{-4}$) enforces consistency between the latent embeddings of the original input and a mildly perturbed version (Gaussian noise level $\sigma = 0.01$). This acts as a smoothness prior on the encoder, encouraging stable latent representations across minor input variations.

Hyperparameters. Full training settings are listed in Table S4.

B.3. Inference Procedure

At inference time, we generate physical parameters by integrating the learned probability-flow ODE using **Heun's second-order method** (a predictor–corrector Runge-Kutta scheme):

$$\textbf{Predictor:} \quad \tilde{\boldsymbol{x}}_{i+1} = \boldsymbol{x}_i + \boldsymbol{v}_\theta(\boldsymbol{x}_i, t_i)\,\Delta t \tag{S3}$$

$$\textbf{Corrector:} \quad \boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \frac{\Delta t}{2}\left[\boldsymbol{v}_\theta(\boldsymbol{x}_i, t_i) + \boldsymbol{v}_\theta(\tilde{\boldsymbol{x}}_{i+1}, t_{i+1})\right] \tag{S4}$$

Table S3. **Joint vs. Separate Multi-Solver.** Joint training unifies three specialized solvers into a single architecture, reducing the total parameter count by **47%** and training time by **53%**. While multi-task regularisation causes minor performance deficits in specific boundary conditions compared to overfitted specialists, the joint model maintains strong average rendering quality (within 0.7 dB).

Training Setup	Params	Train Time	MPM PSNR (↑)		LBS PSNR (↑)		SM PSNR (↑)		Avg. PSNR
			α=0	α=1	α=0	α=1	α=0	α=1	
Separate (3 specialists)	312M	3.2 days	28.83	33.80	36.27	40.35	33.90	41.30	35.7
Joint (UNIPIXIE, Ours)	**167M**	**1.5 days**	**29.25**	32.87	33.83	**41.63**	33.88	38.51	35.0

Table S4. **Training Hyperparameters.** All models trained on 4× NVIDIA A6000 GPUs.

Hyperparameter	Value
Batch size	1 per GPU (effective: 4)
Optimiser	AdamW
Learning rate	5×10^{-5}
Weight decay	0.01
LR scheduler	Cosine annealing (warmup: 3000 steps)
Training epochs	100
Gradient clipping	1.0
Mixed precision	FP16
λ_{LBS}	0.8
$\lambda_{\text{SM}}^{(k)}$	0.5
$\lambda_{\text{SM}}^{(\eta)}$	0.8
λ_{KL}	10^{-4}
Flow inference steps N	15 (Heun's method)
# Latent tokens L	64
Latent dimension C	512
Voxel resolution	64^3
CLIP feature dim D	768

with step size $\Delta t = 1/N$, N=15 steps, and $\boldsymbol{x}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. This requires $2N = 30$ network function evaluations (NFE) and provides a substantially better quality-compute trade-off than the standard Euler integrator at the same NFE budget.

B.4. Ablation: Joint vs. Separate Multi-Solver Training

A natural design question is whether joint training across all three physics solvers is beneficial, or whether three independently trained specialist models would achieve equal or better performance. We answer this empirically in Table S3.

Setup. The *Separate* configuration trains three independent models, each with the same Grid Encoder + FMT decoder architecture as UNIPIXIE, but trained exclusively on one physics solver. The *Joint* configuration is our full UNIPIXIE, where one shared encoder is trained simultaneously with three solver-specific decoders. All other hyperparameters are identical.

Analysis. Rather than optimizing for isolated physics spaces, the joint model necessitates balancing competing gradient signals across all three solvers (MPM, LBS, and SpringMass). As observed in the LBS and SM conditions—which rely on a more restrictive subset of 10 objects compared to the 41 MPM objects—the separate specialists can beneficially overfit to their narrow training distributions, yielding locally higher peak PSNRs (e.g., in SM at α=1). However, the unified model avoids such isolated overfitting. Ultimately, cross-solver supervision establishes a robust shared representation, delivering a **47% reduction in total parameters** and a **53% reduction in training time** for only a marginal 0.7 dB average PSNR trade-off.

C. Baseline Implementation Details

To ensure a fair and reproducible evaluation, we carefully adapted or re-trained all baseline methods on PIXIEMULTIVERSE. This section provides full implementation details for each baseline.

C.1. Deterministic Baselines

PIXIE [5]. PIXIE is our primary deterministic baseline and direct predecessor.

- **Re-training.** We conducted a full re-training of the official PIXIE codebase on PIXIEMULTIVERSE. Since PIXIE is a single-output regression model, we use the *midpoint* of each annotated property range (*i.e.*, the value at $\alpha = 0.5$) as the single ground-truth target. This gives PIXIE a fair advantage at $\alpha = 0.5$-equivalent evaluation points.
- **Hyperparameters.** We used the official open-source implementation and followed the hyperparameter settings reported in the original paper, including the U-Net architecture, learning rate, and optimizer settings, to ensure a faithful comparison.

NeRF2Physics [11] and PUGS [9]. These methods leverage Vision-Language Models (VLMs) for zero-shot physics prediction. While PUGS originally supports continuous material properties (e.g., density, Young's modulus), it does not natively predict discrete material IDs required for physics simulation. To obtain material IDs, we extended PUGS by creating a specialized prompt that instructs GPT-4V to classify materials into 7 discrete categories based on their physical behavior: jelly (0) for deformable materials like rubber and elastic bands, metal (1) for metallic materials, sand (2) for granular materials, visplas (3) for viscoelastic

Table S5. **VLM-Feedback Baseline vs. UNIPIXIE — MPM Solver.** UNIPIXIE is 128× faster with 114× lower $\log E$ MSE, demonstrating that flow matching distills physical knowledge far more efficiently than iterative VLM prompting. ✓ = supported natively; ✗ = not supported.

Method	Runtime ↓	$\log E$ MSE ↓	PSNR ↑	Multi-Solver?
VLM-Strong	1551.1s	1.038	27.7	✗
UNIPIXIE (Ours)	**12.1s**	**0.009**	**30.8**	✓

plastics like clay and putty, fluid (4) for liquids, snow (5) for snow and ice-like materials, and stationary (6) for rigid, non-deformable materials. The prompt asks the VLM to analyze the input image and output a JSON response containing material names paired with their corresponding material IDs (0-6), enabling downstream physics simulation with appropriate material models.

C.2. Generative Ablation Baseline

3D U-Net (Ablation). To validate the superiority of our FMT decoder architecture, we train a conditional 3D diffusion U-Net as a direct ablation.

- **Architecture.** The U-Net takes the concatenated noisy material grid and projected CLIP visual features as input. It follows a standard encoder-decoder structure with residual blocks and cross-attention layers at the two lowest spatial resolutions. The encoder and decoder each use 4 resolution levels with channel multipliers $[1, 2, 4, 8]$.
- **Conditioning.** The control parameter α is embedded via sinusoidal encoding and used to modulate all residual blocks via Adaptive Group Normalisation (AdaGN).
- **Training.** Unlike UNIPIXIE's flow-matching objective, this baseline is trained as a DDPM [3] with a linear noise schedule ($T = 1000$ steps).
- **Inference.** We use the DDIM [10] sampler with 50 steps ($\eta = 0$) to generate material fields. The 50-step DDIM budget is substantially higher than our 30-NFE Heun integration, favouring the U-Net baseline for a fair quality comparison.

C.3. Specialised Solver Baselines

Vid2Sim [1] (LBS Solver). We use the official Vid2Sim implementation for the LBS comparison.

- **Vid2Sim (full).** Standard test-time optimisation for the number of iterations specified in the original paper (∼300 iterations), targeting best reconstruction fidelity. Runtime: ∼521s per object.
- **Vid2Sim (fast).** Accelerated variant running ⅓ of the full iteration count for a more practical runtime comparison (∼140s per object). This variant still substantially exceeds UNIPIXIE's 21.6s three-solver inference time.

Spring-Gaus [12] (Spring-Mass Solver). We compare against two variants of the official Spring-Gaus implementation.

- **Spring-Gaus.** The original model with default hyperparameters, running full test-time optimisation (∼4375s per object).
- **Spring-Gaus (tuned).** To construct the strongest possible Spring-Mass baseline, we perform a hyperparameter search over spring initialisation density, stiffness regularisation weight, and damping coefficient on a held-out validation split of PIXIEMULTIVERSE. The tuned settings improve reconstruction fidelity without changing the number of optimisation iterations. Consequently, the runtime is identical to the default variant (∼4375s), as the computational cost depends only on iteration count and scene complexity.

C.4. VLM-Feedback Baseline (VLM-Strong)

Motivation. A plausible alternative to our supervised feed-forward approach is to leverage a powerful VLM in a test-time feedback loop — iteratively refining material property proposals by observing the resulting MPM simulation. We implement this **VLM-Strong** baseline to directly assess whether such a system can approach the accuracy and efficiency of UNIPIXIE. This baseline directly addresses the reviewer concern of whether a "strong prompting-based VLM baseline with simulation feedback" could achieve comparable results.

Implementation. **Gemini-3.0-Flash** serves as the proposal model. The pipeline proceeds as follows:

1. Multi-view renders and a semantic category label are passed to Gemini-3.0-Flash, which proposes an initial set of per-part material property query ranges (minimum and maximum).
2. MPM simulations are executed using the minimum, maximum, and midpoint of the proposed ranges as three distinct parameter sets, producing three rendered videos.
3. The rendered videos are passed back to the model alongside the previous property estimate with the prompt:
   "Given the simulation results, refine the material properties to improve physical plausibility. Focus on correcting incorrect stiffness or density values."
4. Steps 2–3 repeat until the VLM's proposed ranges produce a successful simulation and the refinement model judges no further modification is needed, or a maximum of 5 iterations is reached.

The pipeline follows a similar structure to our annotation process (CLIP-based segmentation, range proposal), but operates fully automatically without human intervention, relying solely on the VLM refinement loop without access to ground-truth labels. This baseline is evaluated exclusively on the MPM solver. Extending it to LBS or Spring-Mass would require separate solver-specific feedback pipelines and is beyond its intended scope.

Table S6. **Per-α Comparison: PIXIE vs. UNIPIXIE — MPM Solver.** PIXIE is trained only on α=0.5 data; its single prediction is applied to all three evaluation targets. UNIPIXIE is evaluated at the corresponding α value. Our model shows consistently strong performance across the full spectrum while the deterministic baseline degrades away from its training point. Best results per column are **bolded**.

Method	$\alpha = 0.0$ **(Soft)**			$\alpha = 0.5$ **(Mid)**			$\alpha = 1.0$ **(Stiff)**		
	PSNR ↑	**SSIM** ↑	**LPIPS** ↓	**PSNR** ↑	**SSIM** ↑	**LPIPS** ↓	**PSNR** ↑	**SSIM** ↑	**LPIPS** ↓
PIXIE [5]	23.16	0.8868	0.1156	30.17	**0.9225**	**0.0542**	26.04	0.9036	0.0928
UNIPIXIE (Ours)	**29.25**	**0.9050**	**0.1122**	**30.43**	0.9198	0.0810	**32.87**	**0.9246**	**0.0915**

Table S7. **Per-α Decoder Architecture Ablation.** Comparison of the FMT decoder against the 3D U-Net across three physical properties and three control values. Lower MSE is better (↓). The FMT achieves 4–5× lower $\log E$ MSE at every operating point; the gap is largest for Poisson's ratio. Best results per column are **bolded**.

Metric	Decoder	$\alpha = 0.0$ **(Soft)**	$\alpha = 0.5$ **(Mid)**	$\alpha = 1.0$ **(Stiff)**
$\log E$ MSE (↓)	3D U-Net (Ablation)	0.0515	0.0410	0.0305
	UNIPIXIE (FMT)	**0.0102**	**0.0091**	**0.0080**
$\log \rho$ MSE (↓)	3D U-Net (Ablation)	0.1582	0.1464	0.1346
	UNIPIXIE (FMT)	**0.0211**	**0.0194**	**0.0177**
ν MSE (↓)	3D U-Net (Ablation)	0.4821	0.4604	0.4387
	UNIPIXIE (FMT)	**0.0288**	**0.0240**	**0.0192**

Results and Analysis. Table S5 reports the comparison. UNIPIXIE is **128× faster** and achieves **114× lower** $\log E$ **MSE** than VLM-Strong. VLM-Strong suffers from two fundamental limitations beyond computational cost:

- **Accuracy ceiling.** Critically, without access to ground-truth physical labels during training, the VLM cannot acquire precise quantitative knowledge of MPM-scale material parameters. For example, distinguishing $E = 10^4$ Pa from $E = 10^5$ Pa from visual appearance alone is ill-posed without paired simulation data. Our model, trained on 1410 objects with verified ground-truth ranges, learns this mapping explicitly.
- **Single-solver constraint.** VLM-Strong is inherently coupled to MPM and cannot generalise to LBS or Spring-Mass without separate, solver-specific feedback pipelines. UNIPIXIE produces all three solver outputs in a single forward pass — a qualitatively different and more versatile capability.

D. Additional Ablation Studies

This section provides fine-grained quantitative analyses that complement the main paper's results, including a per-α breakdown of the PIXIE comparison and a detailed decoder architecture ablation.

D.1. Detailed Comparison with PIXIE across the Physical Spectrum

Table 2 of the main paper reports UNIPIXIE's performance averaged across $\alpha \in \{0.0, 0.5, 1.0\}$. For convenience of self-contained reading, we reproduce the MPM-solver rows in Table S6 and provide the per-α breakdown, making explicit how the two models compare at each operating point.

Key observations. PIXIE, trained exclusively on α=0.5 data, peaks at the midpoint (30.17 dB PSNR) but degrades substantially at both boundary states (23.16 dB at α=0; 26.04 dB at α=1). This degradation is expected: PIXIE has no mechanism to control stiffness, so its single prediction is systematically mismatched to the ground-truth dynamics at non-midpoint α values. In contrast, UNIPIXIE maintains robust PSNR across the entire spectrum, achieving *higher* values than PIXIE at both boundary states (+6.1 dB at α=0; +6.8 dB at α=1) while remaining competitive at the midpoint (30.43 vs. 30.17 dB). This demonstrates that learning a continuous distribution does not sacrifice accuracy at any single operating point while simultaneously enabling controllable generation across the full physical spectrum.

D.2. Ablation Study: FMT vs. 3D U-Net Decoder

Table 1 of the main paper reports the U-Net ablation averaged across $\alpha \in \{0.0, 0.5, 1.0\}$. Table S7 provides the full per-α and per-property breakdown.

Key observations. The FMT decoder outperforms the U-Net consistently and substantially across all properties and all α values. The gap is most pronounced for Young's Modulus ($\log E$), the most critical stiffness parameter: the FMT achieves **4–5× lower MSE** at every operating point. We attribute this to the Cross-Attention mechanism in each FMT block, which lets the decoder attend globally to the physics-aware latent tokens $\boldsymbol{z}_{\text{latent}}$ at every layer, whereas the U-Net's local convolutional operations limit long-range context propagation. The U-Net's ν MSE is especially poor (0.46 vs. 0.024 at α=0.5). We note that ν occupies a narrow physical range (0.15–0.45), making it highly sensitive to minor prediction errors. We hypothesize that the U-Net's spatial convolution operations inherently blur the sharp, discontinuous material boundaries between different object parts. This

spatial blurring may lead to excessive, unphysical interpolation errors at part-to-part interfaces, degrading ν accuracy. In contrast, the FMT's token-driven cross-attention maintains crisp, part-aware material properties without spatial blurring.

E. Limitations and Broader Impacts

Limitations. While UNIPIXIE successfully generates a controllable continuum of physical properties, it currently conditions on a single scalar α, interpolating along a 1D path between soft and stiff boundaries. Real-world materials reside on complex, multi-dimensional manifolds (*e.g.*, an object could be highly elastic but possess high friction). Future work could explore replacing α with a disentangled latent space to allow for independent control over orthogonal physical dimensions. Additionally, our vision-based formulation inherently leaves interior, occluded structures observation-starved. However, our flow-matching decoder mitigates this by acting as a *volumetric prior*: conditioned on global latent tokens, it infers plausible interior properties from surrounding visible context (*e.g.*, assigning a tree's occluded core properties consistent with its visible bark). This ensures physical coherence under articulation, which could be further improved via active interaction or multi-modal sensing (*e.g.*, tactile feedback).

Broader Societal Impacts. By enabling fast, generalizable, and controllable physics-from-vision, our work democratizes the creation of interactive 3D environments. This has positive implications for AR/VR, video game design, and the generation of realistic simulation environments for training embodied AI agents. We do not foresee immediate negative societal impacts, as the framework models fundamental continuum physics rather than synthesizing sensitive personal data or misleading digital content.

References

[1] Chuhao Chen, Zhiyang Dou, Chen Wang, Yiming Huang, Anjun Chen, Qiao Feng, Jiatao Gu, and Lingjie Liu. Vid2sim: Generalizable, video-based reconstruction of appearance, geometry and physics for mesh-free simulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26545–26555, 2025. 3, 5, 7

[2] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first International Conference on Machine Learning*, pages 12606–12633, 2024. 5

[3] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*. Curran Associates Inc., 2020. 7

[4] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier J. Hénaff, Matthew M. Botvinick, Andrew Zisserman, Oriol Vinyals, and João Carreira. Perceiver IO: A general architecture for structured inputs & outputs. In *International Conference on Learning Representations*, 2022. 4

[5] Long Le, Ryan Lucas, Chen Wang, Chuhao Chen, Dinesh Jayaraman, Eric Eaton, and Lingjie Liu. Pixie: Fast and generalizable supervised learning of 3d physics from pixels. *arXiv preprint arXiv:2508.17437*, 2025. 1, 6, 8

[6] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023. 5

[7] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4172–4182, 2023. 4, 5

[8] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021. 4

[9] Yinghao Shuai, Ran Yu, Yuantao Chen, Zijian Jiang, Xiaowei Song, Nan Wang, Jv Zheng, Jianzhu Ma, Meng Yang, Zhicheng Wang, Wenbo Ding, and Hao Zhao. Pugs: Zero-shot physical understanding with gaussian splatting. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4478–4485, 2025. 6

[10] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022. 7

[11] Albert J. Zhai, Yuan Shen, Emily Y. Chen, Gloria X. Wang, Xinlei Wang, Sheng Wang, Kaiyu Guan, and Shenlong Wang. Physical property understanding from language-embedded feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 28296–28305, 2024. 6

[12] Licheng Zhong, Hong-Xing Yu, Jiajun Wu, and Yunzhu Li. Reconstruction and simulation of elastic objects with spring-mass 3d gaussians. In *European Conference on Computer Vision*, pages 407–423. Springer, 2024. 3, 5, 7