DOUGHMASTER: DEMONSTRATE ONCE, THEN CALIBRATE AND SCALE DOUGH MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Robot demonstrations are expensive, and simulation is a popular way to multiply them. This works well for rigid objects, but not for dough. Every press permanently changes the dough, so a simulator with the wrong material parameters drifts further from reality at every step of a long task. Calibrating the simulator usually requires extra data, such as dedicated probing motions or random exploration. We show that this extra data is unnecessary: the task demonstrations already contain everything needed to identify the material. Our key idea is to use the same demonstrations twice. We first replay them in a differentiable simulator to identify the physical parameters of the dough, and then generate many new demonstrations in the simulator they calibrated. To make identification work with real RGB-D data, we compare the simulation only with the surface each camera can see, and we supervise how the shape changes rather than its absolute position. The calibrated simulator then creates physically consistent variations in dough placement, camera pose, and appearance. On a real dual-arm robot shaping dough into letters, our method identifies material parameters that reproduce the observed deformations more accurately than a DPSI-style baseline using the same data (4.78 vs. 5.10 mm Chamfer distance). With only two teleoperated demonstrations per letter, our augmented data trains a VLA policy that reaches a 54.4% success rate, compared with 24.4% when trained on five times as many real demonstrations.

1 INTRODUCTION

Vision-language-action (VLA) models have made generalist robot manipulation a realistic goal (Zitkovich et al., 2023; Kim et al., 2024; Fang et al., 2026), but their competence remains tightly coupled to the quantity and diversity of demonstrations they are trained on (Khazatsky et al., 2024). This data bottleneck is especially severe for deformable, contact-rich tasks, where even a single demonstration can require minutes of sustained bimanual teleoperation.

Simulation offers a way out of this data bottleneck: a handful of real demonstrations can be multiplied into thousands by re-targeting them to new object layouts (Mandlekar et al., 2023; Garrett et al., 2024), re-rendering them under new viewpoints and appearances (Tobin et al., 2017; Yu et al., 2025), or replaying them in reconstructed digital twins (Torne Villasevil et al., 2024). These pipelines, however, rest on a tacit assumption: object states can be fully described by poses. For rigid objects, transformed trajectories remain physically plausible, and the simulator's material parameters exert negligible influence. This assumption breaks down for deformable, elasto-plastic objects. When a robot manipulates a lump of dough to form a letter, the object state is no longer merely a pose but its entire deformation history. Each contact permanently modifies the material, which then mediates subsequent contacts. Augmenting such demonstrations consequently requires more than rigid object re-positioning: the simulator must faithfully reproduce how *this particular* dough yields, flows, and retains its shape. Under generic material parameters, simulated dough gradually diverges from its real-world counterpart. Since plastic deformation accumulates over time, this divergence compounds across long-horizon tasks, turning augmented simulation data from an asset into a liability. Meanwhile, real demonstrations remain scarce: shaping even a single letter requires minutes of sustained, bimanual, contact-rich teleoperation.

How, then, can we obtain a simulator faithful to a specific real-world material? Existing solutions all require interaction data collected *for its own sake*. System-identification techniques excite the

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

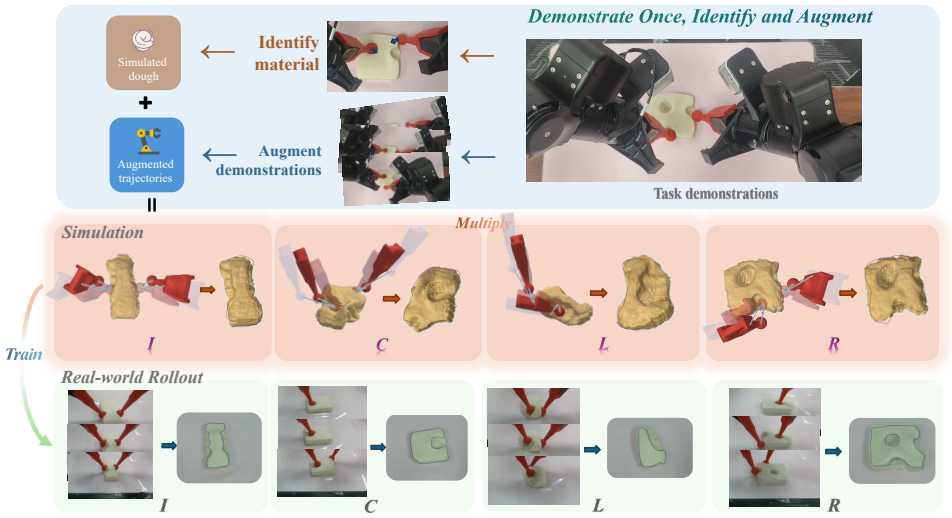


Figure 1: **Demonstrate once, identify and augment.** The same teleoperated demonstrations are used twice: to identify the dough's material parameters in a differentiable MPM simulator, and then to generate augmented trajectories in the identified simulator (middle: letters I, C, L, R). A policy trained on the augmented data is deployed on the real robot (bottom).

material using purpose-designed probing motions, such as brief pokes and pushes (Yang et al., 2025), or actively optimized exploratory trajectories (Memmel et al., 2024). Learned dynamics models, on the other hand, demand dedicated exploration datasets. RoboCook, for example, gathers roughly twenty minutes of random interactions with human-initiated resets for each individual tool (Shi et al., 2023). Beyond introducing an extra, separate data-collection pipeline, these probing interactions sample a behavioral regime distinct from that encountered during target tasks: a one-second poke reveals very little about dough after dozens of folding, flattening, and material-redistributing strokes.

Our key observation is that the data needed for identification are already present in the demonstrations. A long-horizon shaping demonstration is itself a rich physical experiment: the robot applies diverse, large-magnitude contacts, and the dough responds with accumulated plastic deformation, which is exactly the behavior the augmented data must reproduce. We therefore use *the same demonstrations twice*: first to identify the material, then to be multiplied in the simulator they calibrated (Figure 1). Concretely, we teleoperate a dual-arm robot to shape five letters, each over several consecutive episodes without resetting the dough, two demonstrations per letter for 42 episodes in total. Replaying the recorded tool motions in a differentiable Material Point Method (MPM) simulator, we fit the elasto-plastic parameters of the dough to the multi-view point clouds captured at episode boundaries. Two design choices make this fit reliable on real sensor data: a camera-visible surface proxy compares the simulation only with what each camera can actually see, and a dual-scale deformation-increment loss supervises shape *changes*, canceling static calibration biases. The identified simulator then turns each demonstration into many physically consistent variants through spatial re-layout, per-episode layout changes along a continuous deformation history, camera perturbation, and appearance randomization.

On a real dual-arm platform, the identified parameters reproduce the observed deformations more accurately than a DPSI-style baseline using the same data (4.78 vs. 5.10 mm Chamfer distance). A MolmoAct2 policy (Fang et al., 2026) trained on augmented data from only two demonstrations per letter reaches a 54.4% real-robot success rate, compared with 24.4% when trained on five times as many real demonstrations, while the augmented data take only a fraction of the time that real teleoperation requires.

In summary, our contributions are: (1) Elasto-plastic material identification directly from task demonstrations, removing the need for dedicated probing and avoiding probing-task distribution mismatch. (2) A robust differentiable MPM fitting pipeline for noisy partial RGB-D inputs, using a camera-visible surface proxy and dual-scale deformation-increment loss to counteract static sensor calibration bias. (3) A physics-aware simulation augmentation pipeline for long-horizon de-

formable manipulation, validated via real-robot dough-shaping to improve policy performance and reduce real-world teleoperation requirements.

2 Related Work

Elasto-plastic object manipulation. Deformable-object manipulation is challenging due to high-dimensional state spaces and complex dynamics (Sanchez et al., 2018; Yin et al., 2021; Zhu et al., 2022). For elasto-plastic materials such as dough and clay, permanent deformation causes modeling errors to accumulate over long-horizon tasks. One research direction learns particle-based dynamics via graph networks for plasticine shaping, dumpling-making, and cookie-preparation (Li et al., 2019; Shi et al., 2022; 2023; Wang et al., 2023; Zhang et al., 2024a), but these models require dedicated exploratory interaction data (e.g., RoboCook collects about 20min of random tool interactions per tool with human-assisted resets). A second direction leverages differentiable soft-body simulators for trajectory optimization and policy distillation (Huang et al., 2021; Lin et al., 2020; 2022a;b; Qi et al., 2022; Li et al., 2023a); however, simulated materials are not calibrated to real-world dough. Recent imitation-learning work on real clay sculpting remains constrained by limited real-world training data (Bartsch et al., 2024a;b). Our work seeks to combine the fidelity of real demonstrations with simulation-driven scalability.

Physical identification of deformable objects. Since material properties of deformable objects can only be inferred from observed deformation, physical identification requires active interaction (Bohg et al., 2017). Differentiable MPM simulators enable gradient-based parameter fitting by minimizing geometry discrepancy between simulation and observations (Hu et al., 2018; 2019; 2020; Genesis Authors, 2024; Sundaresan et al., 2022; Antonova et al., 2022). Vision-driven approaches recover physical parameters from multi-view videos or video generative priors (Li et al., 2023b; Cai et al., 2024; Zhong et al., 2024; Jiang et al., 2025a; Xie et al., 2024; Zhang et al., 2024b). Yet they primarily target elastic materials with short, reversible motions. Closely related, DPSI (Yang et al., 2025) estimates elasto-plastic parameters using short, manually-designed poking motions, and ASID (Memmel et al., 2024) further optimizes exploratory trajectories. In these prior methods, material identification constitutes a separate stage requiring dedicated excitation actions. By contrast, we perform material identification directly from task demonstrations, eliminating a distinct probing stage.

Simulation-based data augmentation for robot learning. Simulation-based scaling of real demonstrations has become a prevalent strategy for visuomotor policy training. Works including MimicGen and its extensions (Mandlekar et al., 2023; Garrett et al., 2024; Jiang et al., 2025b), CyberDemo (Wang et al., 2024), DemoGen (Xue et al., 2025), Real2Render2Real (Yu et al., 2025), and RialTo (Torne Villasevil et al., 2024) generate or replay demonstration trajectories, while domain randomization (Tobin et al., 2017; Peng et al., 2018) and generative image augmentation (Yu et al., 2023; Chen et al., 2023) enrich training diversity. Nearly all these techniques assume rigid or articulated objects fully described by poses, under which transformed trajectories remain physically plausible. This assumption fails for plastic materials, where state depends on the complete deformation history; existing deformable simulators (Lin et al., 2020; Lu et al., 2024) lack calibration toward specific real materials. Our pipeline augments real-world dough demonstrations inside a simulator whose material parameters are fitted on those very demonstrations, varying spatial arrangement, camera viewpoints, and appearance while enforcing physical consistency with the real material.

3 Method

Our pipeline consists of three stages (Figure 2). (i) **Teleoperation data collection** (§3.1): a dual-arm robot is teleoperated to shape dough into target letter configurations while multi-view RGB-D observations are recorded. (ii) **Physical parameter identification** (§3.2): the recorded tool motions are replayed in a differentiable MPM simulator, and the material parameters of the dough are identified by jointly minimizing a surface-distance loss and a deformation-increment loss. (iii) **Simulation-based data augmentation** (§3.3): the calibrated simulator expands each demonstration with spatial, camera, and appearance variations, and the resulting data are used to train a learning-based policy.

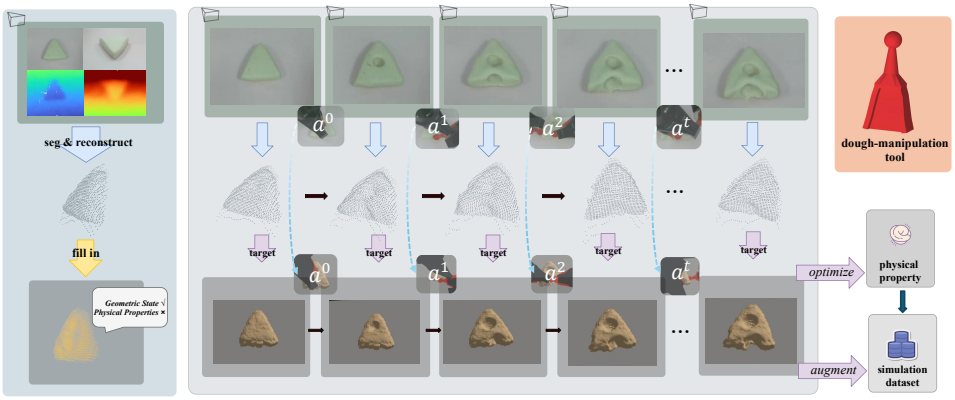


Figure 2: **Material identification from task demonstrations.** Left: the first boundary frame is segmented and back-projected into a surface point cloud, which the waterfall fill converts into volumetric MPM particles (geometry known, material unknown). Middle: the recorded tool actions $a^0, \ldots, a^t$ are replayed in simulation, and the simulated dough at each episode boundary is compared with the observed point cloud (target). Right: the material parameters are optimized through the differentiable rollout, and the identified simulator is then used to generate the augmented dataset.

3.1 Teleoperation Data Collection

Hardware. Each Piper arm is equipped with a dedicated manipulation tool rigidly secured in its gripper; the tool has a trapezoidal rear mounting segment and a rounded tip as the main contact surface for the dough. Two mounted RGB-D cameras (*head*, *forth(front view)*) provide synchronized third-person observations, and two wrist-mounted cameras yield egocentric views.

Multi-episode sequences. We define a *shape* as a single target letter configuration. Within a shape, the robot manipulates the same dough piece across several consecutive *episodes* without resetting it, allowing plastic deformation to accumulate across episode boundaries; each new shape begins with a fresh dough piece. We collect five shapes (A, I, C, L, R), each requiring a different number of episodes (Appendix B.2). For each episode, we record dual-arm joint feedback trajectories synchronized with RGB-D streams. At the beginning and end of each episode, the arms are briefly retracted so that both fixed cameras can capture an unoccluded view of the dough. These *boundary frames* serve as supervision for parameter identification and as geometric references for augmentation.

Point cloud preprocessing. In each boundary frame, the dough is segmented from the RGB image using SAM3, and valid depth pixels inside the mask are back-projected to world coordinates using the per-episode intrinsics and fixed extrinsics. Isolated outliers are then removed with a robust neighborhood filter (Appendix B). The head and forth views are processed independently, yielding one observed point cloud $\mathbf{P}_{s,t,c}$ per shape s, boundary step t, and camera $c \in \{\text{head}, \text{forth}\}$.

3.2 Dough Physical Parameter Identification

3.2.1 Problem Formulation

We model the dough with the Material Point Method (MPM) and a viscoplastic elastoplastic constitutive law (Appendix A), parameterized by $\boldsymbol{\theta} = (E, \nu, \rho, Y, \eta) \in \Theta$, where E is the Young's modulus, ν the Poisson's ratio, ρ the density, Y a constant yield stress, η a viscoplastic viscosity, and Θ the box of admissible parameter values. Let $\mathbf{x}_{s,t}$ denote the full MPM state of shape s at step t, comprising particle positions, velocities, deformation gradients, and APIC affine matrices. The discrete dynamics are

$$\mathbf{x}_{s,t+1} = \mathcal{F}_{\boldsymbol{\theta}}(\mathbf{x}_{s,t},\, \mathbf{u}_{s,t}), \tag{1}$$

where $\mathbf{u}_{s,t}$ is the tool pose obtained from the recorded joint feedback through forward kinematics. Within a shape, the state is propagated *continuously* across episode boundaries, so the simulation retains the complete plastic deformation history. Let $\mathcal{O}_s$ be the set of supervised (time step, camera) pairs $\{(t, c)\}$ of shape s, and let $\mathcal{L}_{s,t,c}$ be the observation loss (defined in §3.2.4). We weight all

shapes equally and average the losses within each shape:

$$\boldsymbol{\theta}^\star = \arg\min_{\boldsymbol{\theta}\in\Theta} \frac{1}{S}\sum_{s=1}^{S} \frac{1}{|\mathcal{O}_s|} \sum_{(t,c)\in\mathcal{O}_s} \mathcal{L}_{s,t,c}(\boldsymbol{\theta}). \tag{2}$$

We denote the objective in equation 2 by $\mathcal{L}(\boldsymbol{\theta})$.

3.2.2 MPM Initialization from Surface Observations

RGB-D sensors observe only the *surface* of the dough, whereas MPM requires a volumetric particle set. We bridge this gap with a *waterfall fill* (Appendix B): the head and forth point clouds of the first unoccluded frame are fused in table-local coordinates, and particles are stacked column by column from the support plane up to the locally observed surface. The resulting $N \approx 3000$ particles are transformed to world coordinates and kept *frozen*, so that all parameter candidates start from an identical initial state. Every particle is initialized with zero velocity, a zero APIC matrix, and an identity deformation gradient.

3.2.3 Camera-Visible Surface Proxy

A camera observes only the *visible* surface of the dough, whereas the MPM state also contains all interior particles. We therefore compare observations against a differentiable *visible-surface proxy* $\mathbf{Q}_{s,t,c} = \mathcal{V}_c(\mathbf{x}_{s,t})$: for camera c at step t, all simulated particles are projected onto the image plane, and within each 2×2-pixel tile only the particle nearest to the camera (i.e., with the smallest depth) is retained. Although the tile-wise selection itself is non-differentiable, the 3D positions of the selected particles remain differentiable with respect to $\boldsymbol{\theta}$ through the MPM forward pass. The loss is thus evaluated on the same visible portion of the surface that the real camera captures.

3.2.4 Observation Loss

For each supervised pair $(t, c) \in \mathcal{O}_s$, the observation loss $\mathcal{L}_{s,t,c} = \mathcal{L}_{\text{surf}} + \mathcal{L}_{\text{def}}$ compares the simulated proxy $\mathbf{Q}_{s,t,c}$ with the observed point cloud $\mathbf{P}_{s,t,c}$ through two complementary terms: $\mathcal{L}_{\text{surf}}$ enforces absolute geometric agreement, and $\mathcal{L}_{\text{def}}$ supervises the deformation relative to the initial frame. For brevity, we drop the indices s, c below and write as $\mathbf{Q}_t$ and $\mathbf{P}_t$.

Robust bidirectional surface distance. Let $d(\mathbf{q}, \mathbf{P}) = \min_{\mathbf{p}\in\mathbf{P}} \|\mathbf{q} - \mathbf{p}\|_2$ denote the distance from a point to a point set, and let $\psi_\sigma(d) = d^2/(d^2 + \sigma^2)$ be a Cauchy-type robust kernel with scale $\sigma = 6\,\text{mm}$. The surface loss symmetrizes the kernelized distance over both matching directions:

$$\mathcal{L}_{\text{surf}} = \frac{1}{2|\mathbf{Q}_t|} \sum_{\mathbf{q}\in\mathbf{Q}_t} \psi_\sigma\big(d(\mathbf{q}, \mathbf{P}_t)\big) + \frac{1}{2|\mathbf{P}_t|} \sum_{\mathbf{p}\in\mathbf{P}_t} \psi_\sigma\big(d(\mathbf{p}, \mathbf{Q}_t)\big). \tag{3}$$

The first term penalizes spurious simulated surface, and the second penalizes observed surface that the simulation fails to reproduce. Since ψ_σ is approximately quadratic for small residuals and saturates for large ones, the loss is robust to occlusions and sensor noise without requiring an explicit inlier mask.

Dual-scale soft-envelope deformation increment. $\mathcal{L}_{\text{surf}}$ is sensitive to static observation biases: a small extrinsic calibration error, for instance, shifts the observed depth by a constant offset and changes the loss even when the simulation is correct. $\mathcal{L}_{\text{def}}$ removes such biases by supervising the *change* of a local depth descriptor relative to the initial frame, which cancels any common additive offset. Consider a point set $\mathbf{P}$ in camera coordinates, where point i has image-plane projection $\mathbf{x}_i^\perp \in \mathbb{R}^2$ and depth z_i. For a grid center $\mathbf{g}$ on the image plane and a scale h, let $a_i(\mathbf{g}) \propto \exp(-\|\mathbf{x}_i^\perp - \mathbf{g}\|^2/2h^2)$ be Gaussian weights normalized so that $\sum_i a_i(\mathbf{g}) = 1$. We define the near-side soft depth envelope as:

$$D_h(\mathbf{P}, \mathbf{g}) = -\tau \log \sum_i a_i(\mathbf{g}) \exp(-z_i/\tau), \qquad \tau = h/3, \tag{4}$$

Intuitively, D_h is a Gaussian-localized soft minimum of the point depths around $\mathbf{g}$ and thus describes the camera-facing surface in that neighborhood.

Writing $\Delta D_{h,t}(\mathbf{P}, \mathbf{g}) = D_h(\mathbf{P}_t, \mathbf{g}) - D_h(\mathbf{P}_0, \mathbf{g})$ for the envelope increment from the initial frame 0 to frame t, we compare the simulated and observed increments through the normalized residual:

$$e_{h,t}(\mathbf{g}) = \frac{1}{h}\big[\Delta D_{h,t}(\mathbf{Q}, \mathbf{g}) - \Delta D_{h,t}(\mathbf{P}, \mathbf{g})\big]. \tag{5}$$

The residual is penalized with the unit Huber loss $H(\cdot)$, and a coarse and a fine scale are combined with equal weights:

$$\mathcal{L}_{\text{def}} = \frac{1}{2} \sum_{h \in \{6,\, 12\}\,\text{mm}} \frac{1}{|\mathcal{G}_{h,t}|} \sum_{\mathbf{g} \in \mathcal{G}_{h,t}} H\big(e_{h,t}(\mathbf{g})\big), \tag{6}$$

where $\mathcal{G}_{h,t}$ contains the grid centers with sufficient point support (at least three points within radius h) in both the initial and the target observed clouds. The coarse scale ($h = 12\,\text{mm}$) captures broad shape changes, while the fine scale ($h = 6\,\text{mm}$) resolves local indentations and concavities. Together, the two losses provide complementary supervision: $\mathcal{L}_{\text{surf}}$ anchors the global shape, and $\mathcal{L}_{\text{def}}$ tracks the deformation relative to the initial configuration.

3.2.5 Differentiable Optimization

We obtain $\nabla_{\boldsymbol{\theta}} \mathcal{L}$ by differentiating through the entire MPM rollout. For a given $\boldsymbol{\theta}$, the simulator replays all tool actions of each shape from the frozen initial state, propagating the state continuously across episode boundaries. The observation loss is evaluated on the GPU at every supervised step; after the forward pass, the adjoint gradients from all supervised steps are accumulated in reverse time through a checkpointed backward pass (Chen et al., 2016). The density ρ enters the dynamics through the particle masses and receives gradients via a differentiable mass initialization.

Because the parameters have different units and span several orders of magnitude, we precondition the gradient with reference magnitudes $\mathbf{r} = (15000,\, 0.47,\, 1200,\, 1800,\, 800)$, ordered as in $\boldsymbol{\theta}$, and take the normalized direction $\mathbf{d} = \mathbf{r} \odot \nabla_{\boldsymbol{\theta}} \mathcal{L} \,/\, \|\mathbf{r} \odot \nabla_{\boldsymbol{\theta}} \mathcal{L}\|_2$. Candidate parameters are then obtained by a projected step

$$\boldsymbol{\theta}_\alpha = \Pi_\Theta\big(\boldsymbol{\theta} - \alpha\, \mathbf{r} \odot \mathbf{d}\big), \qquad \alpha \in \{0.01,\, 0.003,\, 0.001\}, \tag{7}$$

where Π_Θ clips each parameter to its bounds. Step sizes are tried in decreasing order, and the first candidate whose five-shape average loss is strictly lower than the current best is accepted. To reduce wall-clock time, the candidates for $\alpha = 0.01$ and 0.003 are evaluated in parallel, and $\alpha = 0.001$ is tried only if both are rejected. The gradient of the accepted candidate is cached and reused in the next iteration. Parameter bounds and simulator settings are given in Appendix A.

3.3 Simulation-Based Data Augmentation

With the calibrated simulator, we expand each teleoperation demonstration into a large and diverse set of physically consistent robot–dough trajectories. Augmentation proceeds in two stages, which vary the physics and the visual observations, respectively.

Spatial augmentation. We apply a rigid transform T to the initial dough configuration and to all tool targets in table-local coordinates, while keeping the robot bases and cameras fixed. T rotates the dough about its footprint centroid and translates it within the table plane, and a fixed-base IK solver adapts the joint trajectories to track the transformed targets. Candidates that fail the IK-feasibility, joint-limit, or minimum-clearance checks are rejected before simulation. For long multi-episode sequences, we further allow an independent layout T_e for each episode e. At each episode boundary, the dough state is transformed

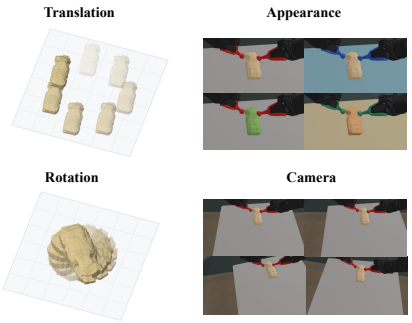


Figure 3: **Augmentation examples.** *Left:* spatial augmentation re-simulates translated or rotated dough and tool targets. *Right:* camera and appearance augmentation re-renders saved states with perturbed camera extrinsics and randomized colors/lighting, without re-running physics.

Table 1: **Physical parameter identification accuracy.** Metrics are averaged over all supervised boundary frames within each shape, then equally weighted across shapes.

Method	$CD_{mm}\downarrow$	$P95_{mm}\downarrow$	$F_{2mm}\uparrow$	$F_{5mm}\uparrow$
Init	7.66	23.87	0.12	0.45
DPSI-on-Genesis	5.10	**15.20**	0.17	0.60
Ours	**4.78**	15.22	**0.19**	**0.63**

Table 2: **Real-robot success rates (%)**, 50 trials per shape. **+Spatial**: spatial layout augmentation; **+Camera**: adds camera extrinsic perturbation; **+App**: adds appearance randomization (full pipeline).

Training data	A	I	C	L	R	Overall
Real only	28	42	12	28	12	24.4
+Spatial	48	58	28	48	32	42.8
+Camera	52	68	42	56	28	49.2
+App (Full, ours)	**58**	**70**	**48**	**58**	**38**	**54.4**

by the incremental transform $\Delta_e = T_e T_{e-1}^{-1}$, under which particle positions, velocities, deformation gradients, and APIC matrices are rotated consistently, and a short transition motion brings the arms to the starting configuration of the next episode. The resulting trajectories shape the dough continuously under varying spatial layouts, all governed by the same calibrated material dynamics.

Camera and appearance augmentation. For each spatially augmented trajectory that passes physics screening, we re-render the saved particle states under multiple visual variants without re-running the physics. The extrinsics of the fixed cameras are jointly perturbed by small rigid transforms (translation≤ 20 mm, rotation$\leq 5^\circ$). Scene appearance is sampled from three tiers: the base configuration with a red tool (60%), mild perturbations of dough color and lighting (30%), and richer material and illumination variations (10%); all tiers share a common 3D table asset rendered with physically based rendering. Appendix C provides rendering details and augmentation statistics.

4 Experiments

On a dual-arm Piper system, we assess (1) identification accuracy vs. a DPSI-style baseline, (2) VLA policy improvement from simulation-augmented data with MolmoAct2, and (3) data-acquisition cost vs. real teleoperation. We test five letter-shaping tasks (A, I, C, L, R), reporting success rates over 50 physical trials per shape per condition.

4.1 Experiment Setup

Tasks. Each task shapes a fresh dough piece (≈ 40 g) into one of five letters. Success is binary, judged by two independent annotators on the final overhead RGB image. **Policies.** We fine-tune *MolmoAct2* on each training set. Inputs are multi-camera RGB images and a language instruction (e.g., *"shape the dough into the letter A"*); outputs are dual-arm joint position targets at 25 Hz.

4.2 Physical Parameter Identification

Protocol. We compare against **DPSI-on-Genesis** (Appendix D): the same Genesis MPM simulator, the same tool motions, initial particles, and parameter bounds, but using a PCD-EMD loss over all volume particles with Adam optimization. Both methods run for 30 iterations from the same initialization. Geometric agreement between the simulated surface and the real target clouds is measured at all supervised episode boundary frames using the metrics of Appendix E.

Results. Table 1 reports symmetric Chamfer distance (CD_{mm}), 95th-percentile surface error ($P95_{mm}$), and F-scores at 2 mm and 5 mm thresholds, averaged equally across the five shapes. Our method achieves 4.78 mm CD versus 5.10 mm for DPSI-on-Genesis, a 6.3% reduction (37.6% relative to the initialization), and improves F_{2mm} from 0.17 to 0.19 and F_{5mm} from 0.60 to 0.63. The two methods are on par in $P95_{mm}$ (15.22 vs. 15.20 mm), indicating that the gain comes mainly from the bulk of the surface rather than the error tail.

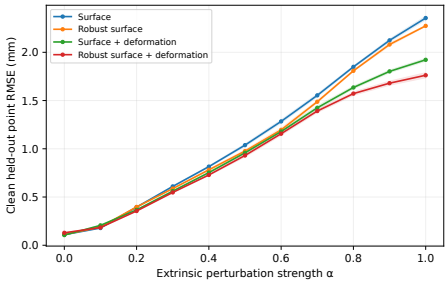
4.3 Controlled Ablations of the Observation Loss

All-target geometry-recovery protocol. We evaluate four observation losses on every real optimization target frame from 10 demonstrations covering the five letters (A1/A2, C1/C2, I1/I2, L1/L2, and R1/R2) giving 74 supervised time points and 148 head/forth point clouds. The four losses are plain surface matching S_{plain}, robust surface matching S_{rob}, and each of these augmented with the relative deformation term D. This is a controlled, geometry-only diagnostic: each candidate is initialized by applying the same prescribed low-dimensional geometric error to its real target template, and three coefficients are then optimized to recover the clean target. The diagnostic therefore evaluates neither reconstruction of unknown shapes nor MPM material identification. All methods share the same preprocessing, initialization, point samples, and budget of 60 Adam updates.

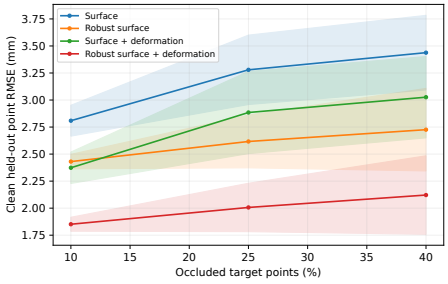
A strength $\alpha \in \{0, 0.1, \ldots, 1\}$ scales a 1° rotation about the world z-axis and a $(3, -2, 1)$ mm translation. This transform is applied about a common pivot computed from the two initial views and perturbs both the initial and the target observations, while the candidate and the clean evaluation geometry remain fixed. The combined test additionally removes a contiguous projection-space region containing 10%, 25%, or 40% of the target points at $\alpha = 1$; the initial references are left unoccluded. We sample up to 640 points per view and hold out one quarter of them for clean correspondence-RMSE evaluation. Means are taken over frames within each sequence, then over sequences within each shape, and finally with equal weight over the five shapes. We report the mean and sample standard deviation of three seed-level aggregates.

Extrinsic bias and incomplete observations. As shown in Figure 4 (a), recovery error grows with the strength of the shared extrinsic perturbation. At full strength, adding D to robust surface matching reduces RMSE from 2.274 to 1.762 mm, which is consistent with relative deformation supervision attenuating time-consistent observation bias. When bias and missing observations are combined (Figure 4 (b)), the full objective $S_{\text{rob}} + D$ attains the lowest aggregate mean at all three occlusion levels. At 25% occlusion, it reaches 2.007 mm, compared with 2.885 mm without the robust penalty ($S_{\text{plain}}+D$) and 2.617 mm without deformation supervision (S_{rob}).

Severe-occlusion stress test. We treat 40% contiguous occlusion as a high-missingness stress condition: removing an entire projected region substantially reduces the available geometric support. Under this condition, the full objective still attains the lowest aggregate error, 2.121 ± 0.363 mm, compared with 3.026 ± 0.376 mm for $S_{\text{plain}} + D$. Its seed-to-seed variation is larger than at 10% occlusion (1.851 ± 0.065 mm), reflecting sensitivity to the sampled points and to the orientation of the missing region. These results establish neither a uniform improvement on every target nor statistical significance. Throughout, the experiment measures geometry recovery within a restricted candidate family, not material-identification accuracy.



(a) Shared extrinsic perturbation



(b) Shared bias with target occlusion

Figure 4: Geometry recovery under the four observation losses (clean held-out RMSE, mm). (a) Shared rigid bias of increasing strength. (b) Full-strength bias with contiguous occlusion of 10–40% of the target points. Surface, Robust surface, and "+ deformation" denote S_{plain}, S_{rob}, and the addition of D.

4.4 Policy Learning with Augmented Data

To understand which components of the augmentation pipeline drive policy improvement, we train MolmoAct2 under a sequence of cumulative ablations: starting from the real teleoperation data alone (**Real only**; 50 demonstrations, i.e., 10 demonstrations per letter). Then based on 2 demonstrations per letter, we successively add spatial layout augmen-

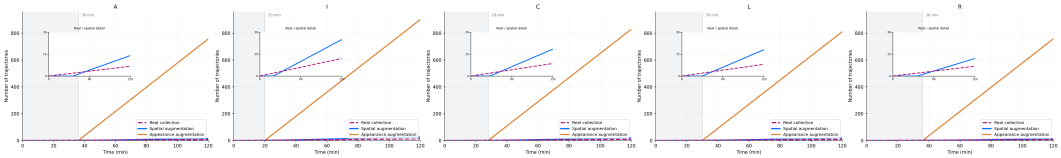


Figure 5: **Data-acquisition efficiency.** Estimated trajectory count over time per letter for real teleoperation, spatial augmentation, and appearance augmentation. Augmentation starts after the two source demonstrations are collected (shaded); insets zoom in on the real and spatial curves.

tation (**+Spatial**), camera extrinsic perturbation (**+Camera**), and finally appearance randomization (**+App**, our full pipeline).

Table 2 reports physical-robot success rates across all five shapes. Training on real demonstrations alone achieves 24.4% overall success. Spatial augmentation of only two demonstrations per letter raises this to 42.8% (+18.4), the largest single gain, by exposing the policy to diverse dough starting positions and tool approach angles. Camera perturbation (+6.4) and appearance randomization (+5.2) contribute further, bringing the full pipeline to 54.4%.

4.5 Data Acquisition Efficiency

Figure 5 compares the number of trajectories obtained over wall-clock time through real teleoperation, spatial augmentation, and appearance augmentation for the five tasks. The mean collection time per valid real trajectory is 18, 10, 14, 15, and 18 minutes for A, I, C, L, and R, respectively. Each estimate is computed from the total collection time required to obtain ten valid trajectories, including setup, unsuccessful attempts, and recollection. These session-level averages also reflect reductions in operator efficiency over prolonged collection, although the plotted linear extrapolation does not explicitly model further fatigue.

Both augmentation curves include the upfront time needed to collect two real demonstrations: 36, 20, 28, 30, and 36 minutes, respectively. Spatial augmentation is modeled at 6, 4, 5, 5, and 7 minutes per additional trajectory, while appearance augmentation is modeled at an aggregate throughput of nine trajectories per minute, assuming saved physical trajectories are available for rendering. For elapsed time t, real collection time r, and spatial augmentation time s, the plotted counts are

$$N_{\text{real}}(t) = \frac{t}{r}, \qquad N_{\text{spatial}}(t) = \frac{[t-2r]_+}{s}, \qquad N_{\text{appearance}}(t) = 9[t-2r]_+,$$

where $[x]_+ = \max(x, 0)$ and augmentation counts exclude the two source demonstrations.

After 120 minutes, the estimated yield is 6–12 real trajectories, 12–25 additional spatial trajectories, or 756–900 appearance variants. Spatial augmentation increases trajectory diversity after the initial collection cost, while appearance augmentation rapidly expands visual diversity. These estimates do not assume linear throughput scaling with GPU count.

5 Conclusion

Deformable elasto-plastic manipulation creates a major data bottleneck for VLA robot policies, as conventional simulation-based augmentation fails without real-world material calibration. We presented an approach that reuses task demonstrations twice: first to calibrate an elasto-plastic simulator, then to generate large-scale physically-consistent augmented trajectories. Our calibration relies on an observation-aware loss with camera-visible surface proxies and deformation-increment supervision, permitting material identification from noisy multi-view RGB-D long-horizon shaping recordings, with no dedicated probing data. Real-robot dough-shaping experiments verify that our calibrated simulator delivers more realistic physical behavior, and policies trained on the augmented data outperform those trained on substantially more real teleoperated demonstrations. Our method has several limitations. The current offline calibration cannot adapt material properties online during execution, and evaluations are limited to dough-shaping tasks. Future work will extend our pipeline to diverse elasto-plastic materials and complex bimanual manipulation scenarios.

AI USE STATEMENT

Generative AI tools were used to assist with language editing and LaTeX formatting.

ETHICS STATEMENT

This work does not involve human subjects or personally identifiable information. Teleoperation data were collected by the authors solely for robot learning research, and all teleoperators provided informed consent. The real-robot experiments use a dual-arm platform to manipulate dough, which poses no significant safety risk; standard safety protocols were followed. We release models and data for research purposes and encourage users to consider potential misuse in physical manipulation tasks.

REPRODUCIBILITY STATEMENT

We provide comprehensive details of our method, including the differentiable MPM simulator, material parameter identification, augmentation pipeline, and policy training, in the main paper and appendix. We will release code, simulated and real demonstration data, model checkpoints, and configuration files. We document hardware setup, hyperparameters, random seeds, and compute resources. Due to hardware-specific constraints, we provide scripts and instructions to reproduce the real-robot experiments.

REFERENCES

Rika Antonova, Jingyun Yang, Krishna Murthy Jatavallabhula, and Jeannette Bohg. Rethinking optimization with differentiable simulation from a global perspective. In *Conference on Robot Learning (CoRL)*, volume 205 of *Proceedings of Machine Learning Research*, pp. 276–286, 2022.

Alison Bartsch, Charlotte Avra, and Amir Barati Farimani. SculptBot: Pre-trained models for 3D deformable object manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 12548–12555, 2024a.

Alison Bartsch, Arvind Car, Charlotte Avra, and Amir Barati Farimani. SculptDiff: Learning robotic clay sculpting from humans with goal conditioned diffusion policy. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7307–7314, 2024b.

Jeannette Bohg, Karol Hausman, Bharath Sankaran, Oliver Brock, Danica Kragic, Stefan Schaal, and Gaurav S. Sukhatme. Interactive perception: Leveraging action in perception and perception in action. *IEEE Transactions on Robotics*, 33(6):1273–1291, 2017.

Junhao Cai, Yuji Yang, Weihao Yuan, Yisheng He, Zilong Dong, Liefeng Bo, Hui Cheng, and Qifeng Chen. GIC: Gaussian-informed continuum for physical property identification and simulation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Qiuyu Chen, Shosuke C. Kiami, Abhishek Gupta, and Vikash Kumar. GenAug: Retargeting behaviors to unseen situations via generative augmentation. In *Robotics: Science and Systems (RSS)*, 2023.

Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

Haoquan Fang, Jiafei Duan, Donovan Clay, Sam Wang, Shuo Liu, Weikai Huang, Xiang Fan, Wei-Chuan Tsai, Shirui Chen, Yi Ru Wang, Shanli Xing, Jaemin Cho, Jae Sung Park, Ainaz Eftekhar, Peter Sushko, Karen Farley, Angad Wadhwa, Cole Harrison, Winson Han, Ying-Chun Lee, Eli VanderBilt, Rose Hendrix, Suveen Ellawela, Lucas Ngoo, Joyce Chai, Zhongzheng Ren, Ali Farhadi, Dieter Fox, and Ranjay Krishna. MolmoAct2: Action reasoning models for real-world deployment. *arXiv preprint arXiv:2605.02881*, 2026.

Caelan Reed Garrett, Ajay Mandlekar, Bowen Wen, and Dieter Fox. SkillMimicGen: Automated demonstration generation for efficient skill learning and deployment. In *Conference on Robot Learning (CoRL)*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2750–2790, 2024.

Genesis Authors. Genesis: A generative and universal physics engine for robotics and beyond. https://github.com/Genesis-Embodied-AI/Genesis, 2024.

Yuanming Hu, Yu Fang, Ziheng Ge, Ziyin Qu, Yixin Zhu, Andre Pradhana, and Chenfanfu Jiang. A moving least squares material point method with displacement discontinuity and two-way rigid body coupling. *ACM Transactions on Graphics*, 37(4):1–14, 2018.

Yuanming Hu, Jiancheng Liu, Andrew Spielberg, Joshua B. Tenenbaum, William T. Freeman, Jiajun Wu, Daniela Rus, and Wojciech Matusik. ChainQueen: A real-time differentiable physical simulator for soft robotics. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6265–6271, 2019.

Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. DiffTaichi: Differentiable programming for physical simulation. In *International Conference on Learning Representations (ICLR)*, 2020.

Zhiao Huang, Yuanming Hu, Tao Du, Siyuan Zhou, Hao Su, Joshua B. Tenenbaum, and Chuang Gan. PlasticineLab: A soft-body manipulation benchmark with differentiable physics. In *International Conference on Learning Representations (ICLR)*, 2021.

Hanxiao Jiang, Hao-Yu Hsu, Kaifeng Zhang, Hsin-Ni Yu, Shenlong Wang, and Yunzhu Li. PhysTwin: Physics-informed reconstruction and simulation of deformable objects from videos. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7219–7230, 2025a.

Zhenyu Jiang, Yuqi Xie, Kevin Lin, Zhenjia Xu, Weikang Wan, Ajay Mandlekar, Linxi Jim Fan, and Yuke Zhu. DexMimicGen: Automated data generation for bimanual dexterous manipulation via imitation learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 16923–16930, 2025b.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, et al. DROID: A large-scale in-the-wild robot manipulation dataset. In *Robotics: Science and Systems (RSS)*, 2024.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *Conference on Robot Learning (CoRL)*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713, 2024.

Sizhe Li, Zhiao Huang, Tao Chen, Tao Du, Hao Su, Joshua B. Tenenbaum, and Chuang Gan. DexDeform: Dexterous deformable object manipulation with human demonstrations and differentiable physics. In *International Conference on Learning Representations (ICLR)*, 2023a.

Xuan Li, Yi-Ling Qiao, Peter Yichen Chen, Krishna Murthy Jatavallabhula, Ming Lin, Chenfanfu Jiang, and Chuang Gan. PAC-NeRF: Physics augmented continuum neural radiance fields for geometry-agnostic system identification. In *International Conference on Learning Representations (ICLR)*, 2023b.

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

Xingyu Lin, Yufei Wang, Jake Olkin, and David Held. SoftGym: Benchmarking deep reinforcement learning for deformable object manipulation. In *Conference on Robot Learning (CoRL)*, volume 155 of *Proceedings of Machine Learning Research*, pp. 432–448, 2020.

Xingyu Lin, Zhiao Huang, Yunzhu Li, Joshua B. Tenenbaum, David Held, and Chuang Gan. DiffSkill: Skill abstraction from differentiable physics for deformable object manipulations with tools. In *International Conference on Learning Representations (ICLR)*, 2022a.

Xingyu Lin, Carl Qi, Yunchu Zhang, Zhiao Huang, Katerina Fragkiadaki, Yunzhu Li, Chuang Gan, and David Held. Planning with spatial-temporal abstraction from point clouds for deformable object manipulation. In *Conference on Robot Learning (CoRL)*, volume 205 of *Proceedings of Machine Learning Research*, pp. 1640–1651, 2022b.

Haoran Lu, Ruihai Wu, Yitong Li, Sijie Li, Ziyu Zhu, Chuanruo Ning, Yan Shen, Longzan Luo, Yuanpei Chen, and Hao Dong. GarmentLab: A unified simulation and benchmark for garment manipulation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, and Dieter Fox. MimicGen: A data generation system for scalable robot learning using human demonstrations. In *Conference on Robot Learning (CoRL)*, volume 229 of *Proceedings of Machine Learning Research*, pp. 1820–1864, 2023.

Marius Memmel, Andrew Wagenmaker, Chuning Zhu, Dieter Fox, and Abhishek Gupta. ASID: Active exploration for system identification in robotic manipulation. In *International Conference on Learning Representations (ICLR)*, 2024.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3803–3810, 2018.

Carl Qi, Xingyu Lin, and David Held. Learning closed-loop dough manipulation using a differentiable reset module. *IEEE Robotics and Automation Letters*, 7(4):9857–9864, 2022.

Jose Sanchez, Juan-Antonio Corrales, Belhassen-Chedli Bouzgarrou, and Youcef Mezouar. Robotic manipulation and sensing of deformable objects in domestic and industrial applications: A survey. *The International Journal of Robotics Research*, 37(7):688–716, 2018.

Haochen Shi, Huazhe Xu, Zhiao Huang, Yunzhu Li, and Jiajun Wu. RoboCraft: Learning to see, simulate, and shape elasto-plastic objects with graph networks. In *Robotics: Science and Systems (RSS)*, 2022.

Haochen Shi, Huazhe Xu, Samuel Clarke, Yunzhu Li, and Jiajun Wu. RoboCook: Long-horizon elasto-plastic object manipulation with diverse tools. In *Conference on Robot Learning (CoRL)*, volume 229 of *Proceedings of Machine Learning Research*, pp. 642–660, 2023.

Priya Sundaresan, Rika Antonova, and Jeannette Bohg. DiffCloud: Real-to-sim from point clouds with differentiable simulation and rendering of deformable objects. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 10828–10835, 2022.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30, 2017.

Marcel Torne Villasevil, Anthony Simeonov, Zechu Li, April Chan, Tao Chen, Abhishek Gupta, and Pulkit Agrawal. Reconciling reality through simulation: A real-to-sim-to-real approach for robust manipulation. In *Robotics: Science and Systems (RSS)*, 2024.

Jun Wang, Yuzhe Qin, Kaiming Kuang, Yigit Korkmaz, Akhilan Gurumoorthy, Hao Su, and Xiaolong Wang. CyberDemo: Augmenting simulated human demonstration for real-world dexterous manipulation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17952–17963, 2024.

Yixuan Wang, Yunzhu Li, Katherine Driggs-Campbell, Li Fei-Fei, and Jiajun Wu. Dynamic-resolution model learning for object pile manipulation. In *Robotics: Science and Systems (RSS)*, 2023.

Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. PhysGaussian: Physics-integrated 3D Gaussians for generative dynamics. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4389–4398, 2024.

Zhengrong Xue, Shuying Deng, Zhenyang Chen, Yixuan Wang, Zhecheng Yuan, and Huazhe Xu. DemoGen: Synthetic demonstration generation for data-efficient visuomotor policy learning. In *Robotics: Science and Systems (RSS)*, 2025.

Xintong Yang, Ze Ji, and Yu-Kun Lai. Differentiable physics-based system identification for robotic manipulation of elastoplastic materials. *The International Journal of Robotics Research*, 44(13): 2126–2155, 2025.

Hang Yin, Anastasia Varava, and Danica Kragic. Modeling, learning, perception, and control methods for deformable object manipulation. *Science Robotics*, 6(54):eabd8803, 2021.

Justin Yu, Letian Fu, Huang Huang, Karim El-Refai, Rares Andrei Ambrus, Richard Cheng, Muhammad Zubair Irshad, and Ken Goldberg. Real2Render2Real: Scaling robot data without dynamics simulation or robot hardware. In *Conference on Robot Learning (CoRL)*, volume 305 of *Proceedings of Machine Learning Research*, pp. 547–577, 2025.

Tianhe Yu, Ted Xiao, Jonathan Tompson, Austin Stone, Su Wang, Anthony Brohan, Jaspiar Singh, Clayton Tan, Dee M, Jodilyn Peralta, Karol Hausman, Brian Ichter, and Fei Xia. Scaling robot learning with semantically imagined experience. In *Robotics: Science and Systems (RSS)*, 2023.

Kaifeng Zhang, Baoyu Li, Kris Hauser, and Yunzhu Li. AdaptiGraph: Material-adaptive graph-based neural dynamics for robotic manipulation. In *Robotics: Science and Systems (RSS)*, 2024a.

Tianyuan Zhang, Hong-Xing Yu, Rundi Wu, Brandon Y. Feng, Changxi Zheng, Noah Snavely, Jiajun Wu, and William T. Freeman. PhysDreamer: Physics-based interaction with 3D objects via video generation. In *European Conference on Computer Vision (ECCV)*, pp. 388–406, 2024b.

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 (ECCV)*, pp. 407–423, 2024.

Jihong Zhu, Andrea Cherubini, Claire Dune, David Navarro-Alarcon, Farshid Alambeigi, Dmitry Berenson, Fanny Ficuciello, Kensuke Harada, Jens Kober, Xiang Li, Jia Pan, Wenzhen Yuan, and Michael Gienger. Challenges and outlook in robotic manipulation of deformable objects. *IEEE Robotics & Automation Magazine*, 29(3):67–77, 2022.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning (CoRL)*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183, 2023.

A Constitutive Model, Simulator Settings, and Parameter Bounds

A.1 Viscoplastic Elastoplastic Constitutive Model

Dough is modeled as a viscoplastic elastoplastic material within the MPM framework. Given material parameters (E, ν), the Lamé coefficients are:

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

At each time step, a trial deformation gradient is computed from the current particle state and decomposed via SVD: $\mathbf{F}_{\text{trial}} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$. The principal logarithmic strains, their deviatoric part, and its norm are:

$$\epsilon_i = \log \max(\Sigma_{ii},\ 0.05), \qquad \mathbf{e} = \boldsymbol{\epsilon} - \tfrac{1}{3}\operatorname{tr}(\boldsymbol{\epsilon})\mathbf{1}, \qquad r = \|\mathbf{e}\|_2.$$

The viscoplastic correction magnitude couples the yield stress Y and viscosity η with the time substep Δt:

$$\Delta\gamma = \max\left(0,\ \frac{2\mu r - Y}{2\mu + \eta/\Delta t}\right).$$

When $\Delta\gamma > 0$ (yielding), the corrected logarithmic strain and updated deformation gradient are:

$$\boldsymbol{\epsilon}' = \boldsymbol{\epsilon} - \frac{\Delta\gamma}{r}\,\mathbf{e}, \qquad \mathbf{F}_{\text{new}} = \mathbf{U}\,\mathrm{diag}(\exp\boldsymbol{\epsilon}')\,\mathbf{V}^\top.$$

Otherwise $\mathbf{F}_{\text{new}} = \mathbf{F}_{\text{trial}}$. No isotropic or kinematic hardening is used (Y is constant), but the plastic deformation history is preserved across time steps through the MPM state variables. The stress contribution to the grid momentum update is:

$$\mathbf{A} = 2\mu\,(\mathbf{F}_{\text{new}} - \mathbf{U}\mathbf{V}^\top)\mathbf{F}_{\text{new}}^\top + \lambda J(J-1)\mathbf{I},$$

where J is the volume ratio maintained by the solver. Particle masses are set as $m_i = \rho V_i$, where $V_i = (0.002\,\text{m})^3$ is the fixed nominal particle volume; ρ participates in the backward pass through differentiable mass initialization.

A.2 Simulator Settings

Setting	Value
Simulator	Genesis differentiable MPM, float64
Time step Δt	0.02 s (32 sub-steps; sub-step $\Delta t = 6.25 \times 10^{-4}$ s)
Grid density	$128\,\text{m}^{-1}$ (grid spacing 7.81 mm)
Nominal particle-volume length	2 mm ($V_i = 8 \times 10^{-9}\,\text{m}^3$)
Waterfall XY/Z spacing	4 mm for identification initializations
Gravity	$(0,\ 0,\ -9.80)\,\text{m s}^{-2}$
Support-surface friction (MPM coupling)	2.00
Tool contact friction	1.00
Tool contact softness	2.00×10^{-3} m
Yield/hardening	Constant Y, no hardening
Backward window	8 physical steps (256 sub-steps) via checkpointing

The backward window controls checkpoint storage and recomputation scheduling; gradients are accumulated across the *full* sequence by injecting adjoint states at each supervised time step in reverse order and propagating them through the checkpointed simulator.

A.3 Material Parameter Bounds

Both our method and the DPSI baseline use the same initial values and box constraints:

Parameter	Initial	Lower	Upper	Unit
E	15,000	10,000	300,000	Pa
ν	0.47	0.01	0.48	—
ρ	1,200	1,000	2,000	kg m^{-3}
Y	1,800	1,000	20,000	Pa
η	800	10^{-6}	5,000	Pa s

Bounds are enforced by per-parameter clipping (Π_Θ) after each update. The initial values are optimization starting points, not prior estimates of the true material; no regularization term pulls parameters toward these values.

Fitted parameters versus production settings. The initialization and bounds above must not be confused with the optimized production values:

Configuration	E (Pa)	ν	ρ (kg/m^3)	Y (Pa)	η (Pa s)
Ours, 30 iterations	13637.06	0.36	1373.11	1701.39	788.65

B Data Preparation Details

B.1 Point Cloud Preprocessing

For each supervised boundary frame, the RGB image (640×360) is segmented using SAM3 with the primary prompt *"a white dough"*, falling back to *"a piece of dough"*, *"a square piece of dough"*, and *"a light green block"* in order. The binary mask is mapped to the original resolution via nearest-neighbor interpolation. Valid depth pixels must satisfy: (i) depth > 50 mm, (ii) pixel coordinates on a stride-2 subsampling grid, and (iii) lie within the segmentation mask. Valid pixels are back-projected to camera-frame 3D points using the per-episode color intrinsics, then transformed to world coordinates by the fixed RGB-D extrinsics. Points outside a 2 cm-padded support-box footprint or below the table plane are discarded.

Isolated-point filtering applies a robust 8-neighbor threshold. Let r_i be the mean 8-neighbor distance of point i, $m = \mathrm{median}(r_i)$, and $a = \mathrm{median}(|r_i - m|)$. Points with $r_i > r_{\max}$ are removed, where:

$$r_{\max} = \max\{3m,\ \ m + \max(3 \cdot 1.48\, a,\ 3.00 \times 10^{-3}\,\mathrm{m})\}.$$

This conservative threshold suppresses sparse noise while preserving real sparse surface details. Each camera is processed independently; the cleaned clouds are saved and frozen before any calibration run.

B.2 Sequence Organization and Supervision Pairs

Episodes within one demonstration are sorted by recording time and treated as a *single continuous sequence*. The MPM state is propagated across episode boundaries without resetting the dough; there is no re-initialization between episodes. A fresh dough piece and a new initial state are used only when switching to a new demonstration.

Supervision is derived from unoccluded episode boundary frames:

- The **first start frame** of each demonstration initializes the MPM geometry (waterfall fill) and serves as the reference $\mathbf{P}_0$ for computing deformation increments ΔD^r. It is *not* added as a separate loss term.
- All **subsequent start frames** and all **episode end frames** contribute to the identification loss $\mathcal{L}$.
- Head and forth cameras provide independent supervision items per frame.

For a demonstration containing n episodes, this convention gives $2n - 1$ paired target times and $2(2n - 1)$ individual camera observations. Consecutive episode end/start frames remain distinct entries. The ten identification demonstrations contain 42 episodes, 74 paired target times, and 148 time–camera observations. They cover three letters, not five distinct letters.

B.3 Waterfall Fill: Algorithm and Statistics

For the nine archived diagnostic initializations, the waterfall fill converts a fused dual-view surface point cloud into a volumetric MPM particle set. All operations are performed in table-local coordinates, where the $+Z$ axis is normal to the support plane.

1. Compute the 2D footprint bounding box of the input points and expand by $r_h = 6$ mm on all sides. Place column centers on a regular 4 mm grid.
2. For each column center $\mathbf{g}$, find all input points within 2D radius r_h. If no points exist, the column is skipped.
3. The local surface top is $z^{\mathrm{top}} = \max_i z_i$ over the neighborhood. Columns with $z^{\mathrm{top}} \le 0$ are skipped.
4. Stack particles at heights $k \cdot 4$ mm for $k = 0, 1, \ldots, \lfloor z^{\mathrm{top}}/4\,\mathrm{mm} \rfloor$.
5. If the gap between z^{top} and the last regular layer exceeds 1.20 mm, add one cap particle at z^{top}.

Table 3: Initialization statistics for all ten demonstrations, read from saved provenance and checked against particle-array lengths.

Demo	Spacing (mm)	XY cols	Valid cols	Regular	Caps	Total N
A1	4.00	783	474	2,742	334	3,076
A2	4.00	780	482	2,765	328	3,093
I1	4.00	612	454	2,569	333	2,902
I2	4.00	1,860	1,465	14,184	977	15,161
C1	4.00	1,071	639	3,806	441	4,247
C2	4.00	975	628	3,823	417	4,240
L1	4.00	915	577	3,756	354	4,110
L2	4.00	1,216	621	3,741	471	4,212
R1	4.00	700	548	2,683	321	3,004
R2	4.00	648	537	2,654	350	3,004

The algorithm assumes a solid column from the support plane to the observed surface top. It cannot reconstruct internal voids, overhangs, or concavities narrower than r_h. Both calibration methods share identical particle sets.

The spacing column denotes both XY column spacing and vertical stacking interval, not solver particle volume. The ten diagnostic initializations use hole radius 6 mm. they were used for identification and geometry diagnostics. "Regular" counts stacked particles across all columns, not the number of vertical layers. Both identification methods use the same frozen particle sets for the five fitted demonstrations.

B.4 Action Resampling and Tool Motion

Joint feedback is read at the native recording rate ($\sim$10–25 Hz) and resampled to the 20 ms physical time axis via per-joint linear interpolation (numpy.interp). Within each episode, extrapolation clamps to the endpoint values; the gap between episodes is held at the preceding episode's final pose. Tool poses are computed at each resampled time step via fixed forward kinematics (URDF) and a fixed tool-mounting transform. The full tool mesh (trapezoidal body with rounded tip) is used for MPM contact via a signed-distance field; no spherical approximation is used.

C Augmentation Details

C.1 Spatial Augmentation: Transforms and Candidate Generation

All spatial transforms are defined in table-local coordinates and parameterized by an in-plane translation (d_x, d_y) and a yaw rotation ψ about the dough footprint centroid $\mathbf{p}$ (with p_z set to zero):

$$D(\mathbf{R}_z(\psi),\, \mathbf{d}) = \begin{bmatrix} \mathbf{R}_z(\psi) & \mathbf{p} - \mathbf{R}_z(\psi)\mathbf{p} + \mathbf{d} \\ \mathbf{0} & 1 \end{bmatrix}, \qquad T = T_{WS}\, D\, T_{WS}^{-1},$$

where T_{WS} is the frozen table-to-world rigid transform. Robot bases, fixed cameras, and the table asset are not moved.

We generate two families of candidates:

Whole-sequence uniform transforms. For each source demonstration, we enumerate $7 \times 7 \times 9 = 441$ combinations of $d_x, d_y \in \{-30, -20, -10, 0, 10, 20, 30\}$ mm and $\psi \in \{-20^\circ, -15^\circ, \ldots, 20^\circ\}$ (step 5°). All episodes share the same transform. With 2 source demonstrations per letter and 5 letters, this yields $441 \times 10 = 4{,}410$ candidates (including 10 identity transforms that reuse the reference simulation).

Per-episode independent layouts. Each episode independently selects one of 8 target positions on a circle of radius 15 mm (spaced 45° apart, no yaw). For n episodes, there are 8^n combinations; we sample 500 per source demonstration using a fixed random seed (deterministic, numpy PCG64). With 2 sources per letter and 5 letters, this yields 5,000 candidates.

At each episode boundary where the layout changes, the dough state is transformed by the incremental update $\Delta_e = T_e T_{e-1}^{-1}$:

$$\mathbf{x}' = \mathbf{Q}\mathbf{x} + \mathbf{t}, \quad \mathbf{v}' = \mathbf{Q}\mathbf{v}, \quad \mathbf{F}' = \mathbf{Q}\mathbf{F}, \quad \mathbf{C}' = \mathbf{Q}\mathbf{C}\mathbf{Q}^\top,$$

where $\Delta_e = [\mathbf{Q} \mid \mathbf{t}]$. Scalar fields ($J_p$, active, hardening_a) are unchanged. For purely in-plane translation ($\mathbf{Q} = \mathbf{I}$), only positions change. A short joint-space bridge transition (5th-order polynomial, ≤ 120 steps) moves the arms to the next episode's starting configuration without moving the dough.

C.2 Pre-Screening

Before running full physics simulation, each candidate is screened:

1. **Footprint clearance.** Transformed initial particles must have ≥ 3 mm margin inside the support-box XY footprint.
2. **IK feasibility.** A fixed-base IK solver must track all transformed tool targets in the interaction interval (defined as the frames where tool-to-dough gap ≤ 40 mm) within 0.5 mm translation and 0.5 rotation error, with per-step joint increments ≤ 0.15 rad.
3. **Tool–dough clearance.** At sampled pre-simulation time steps, the transformed tool convex hull must maintain ≥ 10 mm clearance from the (reference-transformed) dough particles.

Candidates failing any check are marked rejected_precheck and skipped.

C.3 Geometry Screening After Simulation

After full MPM simulation, each candidate is geometrically screened by comparing its end-of-episode particles (after inverting the spatial transform) against the source reference simulation:

$$d_{\text{sym}} = \tfrac{1}{2}\Big[|P|^{-1} \sum_{\mathbf{p}\in P} \min_{\mathbf{q}\in Q} \|\mathbf{p}-\mathbf{q}\|_2 + |Q|^{-1} \sum_{\mathbf{q}\in Q} \min_{\mathbf{p}\in P} \|\mathbf{q}-\mathbf{p}\|_2\Big].$$

A candidate passes (passed) if, for every episode: $d_{\text{sym}} \leq 2$ mm, centroid deviation ≤ 5 mm, and per-axis size ratio within $[0.85, 1.15]$. Candidates completing simulation but exceeding thresholds are flagged needs_review and retained for manual inspection.

C.4 Camera and Appearance Augmentation

For each passed spatial candidate, we generate 10 visual variants by re-rendering the saved particle states without re-running any physics. The 10 variants are split across three tiers:

- **Base-red (6/10).** Original camera configuration; tool rendered in red (#C83C32); current dough color and lighting.
- **Mild variation (3/10).** Tool still red; dough hue shifted ± 10, saturation scaled $[0.85, 1.15]$, value scaled $[0.9, 1.1]$; ambient and directional light intensity independently scaled $[0.85, 1.15]$; main-light direction perturbed ≤ 5. Fixed cameras jointly perturbed by a rigid transform (translation ≤ 20 mm, rotation ≤ 5).
- **Rich variation (1/10).** Tool and dough colors sampled from predefined palettes; table/mat hue shifted ± 15; lighting scaled $[0.7, 1.3]$, direction perturbed ≤ 15; fixed cameras perturbed as above plus independent per-camera residuals (translation ≤ 5 mm, rotation ≤ 1); focal length scaled $[0.97, 1.03]$.

Rendering uses a shared 3D table asset with physically based materials. Dough surface geometry is reconstructed per frame from MPM particles via marching cubes (grid spacing 0.75 mm, Gaussian smoothing $\sigma = 1.5$ cells) and cached by particle-state hash, so multiple visual variants of the same physical trajectory reuse the same mesh. All randomization is deterministic: variant seeds are derived from the candidate ID and variant index.

D DPSI BASELINE ADAPTATION

We compare against a DPSI-style baseline adapted to the same Genesis MPM simulator, data, and experimental protocol. We refer to it as **DPSI-on-Genesis**.

D.1 RETAINED LOSS AND OPTIMIZER

The baseline uses a PCD-EMD objective. Let $\mathbf{P}$ be the real point cloud after 5 mm voxel downsampling and $\mathbf{X}$ all simulated volume particles ($|\mathbf{P}| \leq |\mathbf{X}|$). An injective assignment $\pi^\star$ from real to simulated points is found by solving a rectangular linear assignment problem (via scipy.optimize.linear_sum_assignment):

$$\pi^\star = \arg\min_{\pi \text{ injective}} \sum_{i=1}^{|\mathbf{P}|} 1000\|\mathbf{p}_i - \mathbf{x}_{\pi(i)}\|_2, \qquad \mathcal{L}_{\text{EMD}} = \sum_{i=1}^{|\mathbf{P}|} 1000\|\mathbf{p}_i - \mathbf{x}_{\pi^\star(i)}\|_2.$$

Note that distances are in millimeters and summed (not averaged) within each observation; inter-observation and inter-shape averaging follows the same protocol as our method (Eq. 2). Parameters are updated by bias-corrected Adam:

$$\boldsymbol{\theta}_{k+1} = \Pi_\Theta\left[\boldsymbol{\theta}_k - \boldsymbol{\ell} \odot \frac{\hat{m}_k}{\sqrt{\hat{v}_k} + 10^{-8}}\right],$$

with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and per-parameter learning rates $\boldsymbol{\ell} = (4000,\ 0.001,\ 10,\ 100,\ 10)$ for (E, ν, ρ, Y, η).

D.2 CONTROLLED COMPARISON PROTOCOL

To isolate the identification design choices (loss function, simulated point set, optimizer), we hold the following elements identical across both methods:

Element	Shared setting
Simulator	Same Genesis MPM, constitutive model, float64 precision
Parameter space	Same 5D $\boldsymbol{\theta} = (E, \nu, \rho, Y, \eta)$; η added to baseline
Initial values & bounds	Identical (Table in Appendix A)
Real data	Same teleoperation recordings, tool motions, initial particles
Episode propagation	Continuous within-shape, no reset between episodes
Supervision pairs	Same 74 time-camera pairs (5 mm downsampled for baseline)
Failure handling	Non-finite values and failed shapes are flagged, not silently dropped

The three key differences are summarized below:

	Ours	DPSI-on-Genesis
Simulated point set	Visible-surface proxy (2×2 px z-buffer)	All volume particles
Real target	Cleaned surface cloud	Same, then 5 mm voxel-downsampled
Loss	$\mathcal{L}_{\text{surf}} + \mathcal{L}_{\text{def}}$ (Cauchy + envelope)	PCD-EMD (mm, summed)
Optimizer	Scaled GD + acceptance	Adam, no line search

E EVALUATION PROTOCOL

E.1 GEOMETRIC METRICS

Both methods are evaluated at their *final-iteration* parameters (30 iterations each) using an identical visible-surface extraction pipeline (same 2×2 px z-buffer proxy used during training) and fixed 2 mm world-grid voxelization of both simulated and real clouds. Let $\mathbf{Q}$ be the simulated visible surface and $\mathbf{P}$ the real target.

Symmetric mean Chamfer distance ($\mathrm{CD}_{\mathrm{mm}}$).

$$\mathrm{CD}_{\mathrm{mm}} = \frac{1000}{2}\left(\mathrm{mean}_{\mathbf{q}\in\mathbf{Q}}\, d(\mathbf{q},\mathbf{P}) + \mathrm{mean}_{\mathbf{p}\in\mathbf{P}}\, d(\mathbf{p},\mathbf{Q})\right),$$

reporting non-squared Euclidean distances in millimeters.

95th-percentile surface distance ($\mathrm{P95}_{\mathrm{mm}}$).

$$\mathrm{P95}_{\mathrm{mm}} = 1000\max\left\{\mathrm{pct}_{95}\big(d(\mathbf{Q},\mathbf{P})\big),\ \mathrm{pct}_{95}\big(d(\mathbf{P},\mathbf{Q})\big)\right\},$$

measuring the worst-case tail of the surface error distribution.

Precision, Recall, and F-score at threshold δ. For $\delta \in \{2\,\mathrm{mm},\ 5\,\mathrm{mm}\}$:

$$\mathrm{Pr}_\delta = \Pr_{\mathbf{q}\in\mathbf{Q}}\big[d(\mathbf{q},\mathbf{P}) \le \delta\big], \quad \mathrm{Re}_\delta = \Pr_{\mathbf{p}\in\mathbf{P}}\big[d(\mathbf{p},\mathbf{Q}) \le \delta\big], \quad F_\delta = \frac{2\,\mathrm{Pr}_\delta\,\mathrm{Re}_\delta}{\mathrm{Pr}_\delta + \mathrm{Re}_\delta}.$$

All metrics are computed per supervised observation, then averaged within each shape, then equally weighted across the 5 shapes. Per-shape and per-camera breakdowns are also reported.