PointZero: 3D Point Track Completion for Learning Transferable 3D Dynamics

Bardienus P. Duisterhof[1] **Kaifeng Zhang**[2]* **Adam Hung**[1]* **Bowen Wen**[3]
Stan Birchfield[3] **Yunzhu Li**[2] **Deva Ramanan**[1] **Jeffrey Ichnowski**[1]
[1]CMU [2]Columbia [3]NVIDIA

pointzero-wm.github.io



(a) **Robot-Free Pre-Training:** Partial Point Tracks ⇒ Dense Point Tracks



(b) **Dynamics Modeling:** Robot-Conditioned 3D Prediction

(c) **Imitation Learning Post-Training:** Predicting Robot Actions

Figure 1: PointZero proposes *3D point track completion* as a pre-training objective for learning rich 3D dynamics priors. Given one RGB-D image and one or a few point trajectories (shown in orange), PointZero predicts the future 3D tracks of all observed scene points. PointZero is a robot-free pre-training objective for instilling 3D dynamics understanding in world models. Post-trained for dynamics modeling [1] and robot manipulation [2], it outperforms *application-specific* baselines.

Abstract

World models endow perceptual systems with the ability to predict how scenes evolve under interaction. They are most beneficial when trained on diverse volumes of data, to instill a rich prior into downstream applications. Existing methods typically require robot action labels to learn action-conditioned 3D dynamics, which excludes web video data from the training pool. We study *3D point track completion* as a pre-training objective for learning transferable 3D dynamics without robot data. Given a single RGB-D observation and sparse partial 3D trajectories (tracks), we predict future 3D tracks of all observed points. We show this objective produces a rich 3D dynamics prior, without requiring robot action labels. We contribute a diverse dataset of 2.9 million synthetic frames spanning deformable, articulated, and rigid objects, and use it to train PointZero. We show that a flexible and expressive transformer, PointZero, outperforms prior methods on the same data. We demonstrate the utility of our pre-training objective by post-training PointZero for two downstream applications: (1) action-conditioned 3D dynamics prediction and (2) imitation learning. When fine-tuned to condition on end-effector pose, PointZero outperforms the baselines on the recent PGND 3D dynamics benchmark. When fine-tuned to predict robot actions and 3D tracks, PointZero outperforms or matches the baselines on 6/7 simulated and real-world robot manipulation tasks. We furthermore evaluate training PointZero from scratch to isolate the benefits of our proposed architecture from those of our proposed pre-training objective and dataset. We release the dataset, checkpoints, and full training recipe.

*Equal contribution.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

1 Introduction

Humans intuitively anticipate how the world responds to interaction: pulling a drawer, folding a sock, or sliding a cup. Endowing perception systems with comparable predictive abilities promises to unlock applications across robotics, simulation, and AR/VR/XR.

Challenges. Learning a generic 3D dynamics model is inherently challenging due to the vast *diversity* of physical phenomena, the *underobservability* of many aspects of physics, and the difficulty of characterizing the space of all possible *interactions*. Despite such challenges, humans readily interact with articulated, rigid, and deformable objects, and can predict plausible dynamics "zero-shot" before interacting with a particular object of interest. Prior methods learn robot-action-conditioned 3D dynamics models by fitting a separate model per scene from hundreds of interactions with the target object [1, 3]. Even hundreds of interactions may be insufficient to learn complex dynamics from scratch, motivating pre-training objectives that provide transferable priors over 3D scene dynamics.

Pre-training. Large-scale pre-training on video data has led to dramatic improvements in data efficiency and generalization capabilities of robot learning systems [4, 5]. However, most such approaches pre-train 2D visual representations or video prediction models, which cannot explicitly represent metric 3D scene motion.

Recent 3D world models have begun scaling dynamics prediction in 3D, but rely on robot interaction data with action labels [6]. Requiring robot action labels makes data collection considerably more expensive and excludes the vast amount of available web video. This raises the question, *can we learn 3D dynamics priors without action annotations?*

3D Point Track Completion. We propose 3D point track completion as a robot-free training objective for learning transferable 3D dynamics. Given a single RGB-D observation and 3D point trajectories for a few sparse points, we predict the 3D point tracks of all observed points. Three properties make the objective scalable. *Robot-free supervision:* The labels and conditioning features are both just 3D point tracks, available from simulation and in principle from any video via point tracking [7, 8]. No robot- or embodiment-specific annotations are needed. *Task-agnostic supervision:* As with other dynamics-modeling objectives, training examples need not correspond to successful task completions. Arbitrary play data or free-form interactions with objects can all provide useful supervision. *Flexible representation:* 3D point tracks can represent arbitrary objects, including those with high-dimensional state such as cloth.

Post-training. We study the utility of our pre-training objective by post-training pre-trained models for two practical applications. First, we adapt the model to condition on robot pose instead of partial point tracks, and train it on scene-specific action-conditioned dynamics modeling using the PGND benchmark [1]. Second, we adapt PointZero for robot manipulation by adding a small head to predict robot actions and supervise with robot manipulation demonstrations.

Zero-shot generalization. We further evaluate whether PointZero learns a *generalizable* dynamics prior by testing the model's zero-shot capabilities. PointZero's architecture consistently outperforms other 3D dynamics-modeling architectures when all methods are trained on our synthetic dataset and evaluated zero-shot on PGND scenes [1] and on a custom test set of unseen real-world objects.

Contributions.

- **Problem formulation:** We introduce 3D point track completion as a flexible and transferable pre-training objective for learning arbitrary 3D dynamics without requiring action labels.
- **Performant task transfer:** When post-trained for *action-conditioned dynamics modeling* and *robot manipulation*, PointZero outperforms strong *application-specific* baselines and training the PointZero architecture from scratch. This suggests that our pre-trained model learns broadly useful 3D dynamics priors that transfer effectively to robotic applications.
- **Dataset:** We release a diverse 4D synthetic dataset with dense per-point trajectory annotations spanning deformable, articulated, and rigid object interactions. We also release the training and inference code for pre- and post-training, as well as pre-trained checkpoints.

2 Related Work

Scene-specific Dynamics Models. A common problem setting in prior work is to fit a dynamics model to interaction data collected in a specific scene. Methods grounded in physics such as mass-spring systems [9, 10], the Finite Element Method (FEM) [11], Position-Based Dynamics (PBD) [12], and the Material Point Method (MPM) [13, 14] perform well only when state estimation and system identification are successful.

The difficulty in properly identifying system parameters under partial observability and noise has led researchers to explore learning-based approaches. Graph-Based Neural Dynamics (GBND) [15, 16, 17] has shown great promise in simulating complex, high-dimensional deformable objects. However, its reliance on manually designed graph topologies and local message passing limits its applicability across a broader range of objects, particularly to rigid and articulated objects where dynamics involve abrupt changes and constrained motion. Several works explored particle-based predictions without relational edges [1, 18, 3, 19] that can learn dynamics models successfully on challenging materials, but come with a limitation that the methods are only trained on single objects or scenes with predefined physical parameter types. In contrast, PointZero combines a simple transformer architecture with a 3D point track completion objective to learn representations that adapt to scene-specific dynamics in post-training, outperforming these baselines on their own benchmarks.

Large-scale 3D Representation Learning. Large-scale pre-training has transformed 3D computer vision, enabling models to generalize across diverse scenes, objects, and viewpoints. DUSt3R [20] and follow-up works [21, 22, 23] have substantially advanced 3D reconstruction, while other efforts have improved object pose estimation [24], stereo depth estimation [25], and scene completion [26].

Dynamics models trained at scale have primarily focused on image next-state prediction, either in latent [27] or pixel [28, 29] space (e.g., video generation). Scalable 3D dynamics learning has remained largely unexplored, owing to the scarcity of large 3D temporal datasets. Recent work explores language-conditioned 3D point forecasting [30] and pose-trajectory completion [31].

Concurrent to our work, PointWorld [6] studies scalable 3D dynamics prediction and can model scene dynamics without scene-specific interaction data or explicit object geometry priors. However, it requires calibrated real-world robot interaction data to train, which limits scalability. PointZero instead pre-trains on synthetic point track completion, conditioning on sparse tracks to predict dense trajectories without requiring any robot annotations.

Pre-training for Robotics. Prior works investigated pre-training visual representations for downstream manipulation policies. One approach is to learn generic visual features using self-supervised learning methods such as masked autoencoding [32, 33] or DINO [34], and then fine-tune on robotics tasks. Rising in popularity is predicting task-conditioned futures [35, 36, 37] across modalities such as video, DINO, point tracks and depth. Other methods learn representations through explicit supervision over manipulation-relevant signals, such as video-language alignment [38], human actions [39], or point trajectories [2, 40]. These works suggest that structuring representation learning around features relevant to manipulation can improve downstream policy learning. Our 3D point track completion objective follows a similar principle: by predicting dense 3D point trajectories, the model is encouraged to learn representations that capture task-relevant object geometry, motion, and dynamics. The closest setting to ours is 3PoinTr [2], which uses task-specific 3D point track prediction as pre-training for downstream manipulation; we compare against 3PoinTr and other baselines in our imitation-learning experiments.

3 Problem Statement

We introduce the pre-training objective of *3D point track completion*.

Point cloud observation. Starting from a single RGB-D observation, we use the RGB image $I \in \mathbb{R}^{H\times W\times 3}$, depth map $D \in \mathbb{R}^{H\times W}$, foreground mask $M \in \{0,1\}^{H\times W}$, and camera intrinsics $K \in \mathbb{R}^{3\times 3}$ to unproject the masked depth into an observed 3D point cloud $P^{\text{obs}} \in \mathbb{R}^{N_p\times 3}$, where N_p is the number of foreground pixels.

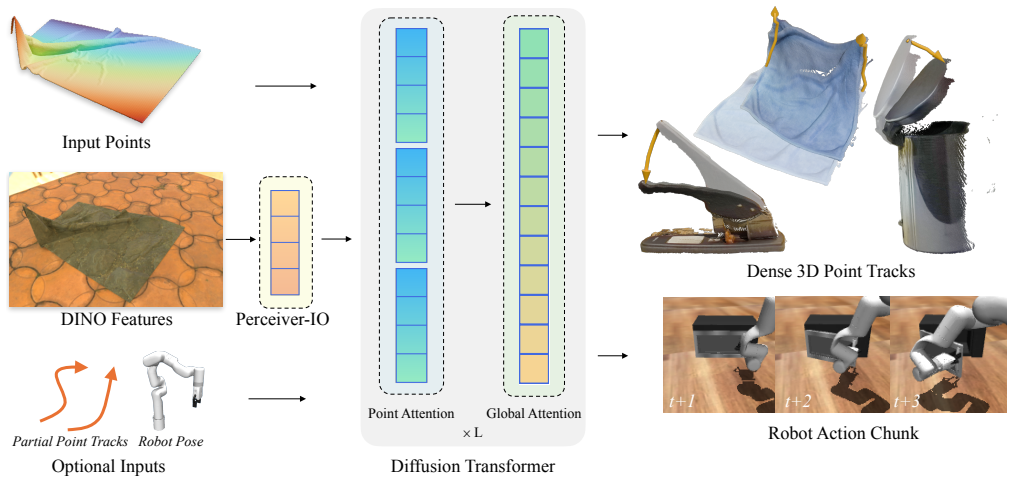

Figure 2: PointZero architecture overview. PointZero takes in a single RGB-D image and a small number of complete point tracks, and predicts dense 3D point trajectories. PointZero can also be fine-tuned to condition on robot pose instead of partial point tracks, or to output robot action chunks in addition to point track predictions. PointZero combines a Perceiver-IO [41] architecture with self-attention and cross-attention to encode diverse dynamics.

Partial point track conditioning. We condition the model on a small number of partial 3D trajectories: we select N_a conditioning points over T timesteps, represented as $A \in \mathbb{R}^{T \times N_a \times 3}$, where empirically $N_a \in \{1, 2, 3\}$.

Prediction. Given the initial observation P^{obs}, visual features extracted from I, and the conditioning A, PointZero completes point tracks $P \in \mathbb{R}^{T \times N_p \times 3}$, with $P_{1,i} = P_i^{\text{obs}}$ and predicted positions at frames $\tau \in \{2, \ldots, T\}$.

The point-level formulation provides a unified dynamics representation for arbitrary object types. The pre-training objective can be used or adapted for several applications, including **partial point track completion:** the trained model directly predicts dense point tracks, given one or more conditioning point tracks. **Post-training for action-conditioned dynamics:** replace the partial point track conditioning with robot end-effector pose conditioning. **Post-training for robot manipulation:** attach a lightweight action prediction head to the pre-trained representation and supervise it with observation-action pairs from expert robot demonstrations.

4 Methods

We propose PointZero, a single-transformer architecture for modeling action-conditioned 3D dynamics. The model combines (i) point cloud geometry, (ii) action trajectories, and (iii) visual features into a single diffusion transformer (DiT) that predicts the target state from noise.

Point tokens. Given the observed point cloud P^{obs}, we repeat each initial position over the $T-1$ future frames to form $r_i \in \mathbb{R}^{3(T-1)}$, and embed it using a shared MLP E_P, yielding $X_P \in \mathbb{R}^{N_p \times d}$.

Partial point track tokens. For each time step $\tau \in \{1, \ldots, T\}$ and trajectory index $j \in \{1, \ldots, N_a\}$, we embed the 3D coordinate $A_{\tau,j} \in \mathbb{R}^3$ using a shared MLP E_A. We inject positional information using a time embedding $e_t(\tau) \in \mathbb{R}^d$, produced by an MLP from the scalar τ, and a trajectory-index embedding $e_a(j) \in \mathbb{R}^d$, looked up from a learned table of N_a vectors. The resulting action token is $(X_A)_{\tau,j} = E_A(A_{\tau,j}) + e_t(\tau) + e_a(j)$, and we flatten these tokens into a single action sequence $X_A \in \mathbb{R}^{(TN_a) \times d}$.

Visual tokens. We extract dense image features using DINOv2 [42], yielding tokens $F_{\text{DINO}} \in \mathbb{R}^{\lfloor H/14 \rfloor \times \lfloor W/14 \rfloor \times d}$. To reduce memory, we compress them into N_V visual tokens using a Perceiver-IO module [41]: starting from learned latent queries $Z \in \mathbb{R}^{N_V \times d}$, we apply L layers of self-attention over Z and cross-attention from Z to F_{DINO}. This produces $X_V = \text{Perceiver}(Z, F_{\text{DINO}}) \in \mathbb{R}^{N_V \times d}$.



Figure 3: We contribute a novel dataset to learn diverse dynamics in simulation and the real world. The dataset contains 2.9 million synthetic image frames with deformable, articulated and rigid objects. We also collect a dataset with 14 objects and 124 interactions in the real world for evaluation of zero-shot sim-to-real transfer.

Diffusion transformer (DiT). We use a diffusion transformer with alternating layers of self-attention and cross-attention. For each point i, the query concatenates the current noisy future trajectory $p_{i,t} \in \mathbb{R}^{3(T-1)}$ with r_i, the initial XYZ position repeated $T-1$ times, and projects the result with a shared MLP $E_Q : \mathbb{R}^{6(T-1)} \to \mathbb{R}^d$. This yields $X_{Q,i} = E_Q(\text{concat}(p_{i,t}, r_i))$, with $X_Q \in \mathbb{R}^{N_p \times d}$; a learned diffusion-time embedding is added to each query.

The cross-attention keys and values are formed by concatenating the point, action, and visual embeddings: $X_K = \text{concat}(X_P, X_A, X_V) \in \mathbb{R}^{(N_p + TN_a + N_V) \times d}$.

4.1 Training Objectives

We supervise the full trajectory per point to provide a rich supervision signal. We index the $T = 10$ physical frames by $\tau \in \{1, \dots, T\}$, with the initial observation at $\tau = 1$, and define the flattened future trajectory $p_i = \text{flatten}(P_{i,2:T}) \in \mathbb{R}^{3(T-1)}$, where $P_i \in \mathbb{R}^{T \times 3}$ denotes the ground-truth trajectory of point i in P. For all losses below, $\|\cdot\|_w^2$ denotes the coordinate-averaged, temporally weighted squared error defined in Appendix A.1; losses are averaged over points and training examples.

Regression. Given the dynamics model g_θ parameterized by θ, we directly regress the trajectory

$$\mathcal{L}_{\text{reg}} = \frac{1}{N_p} \sum_{i=1}^{N_p} \left\| g_{\theta,i}(P^{\text{obs}}, A, I) - p_i \right\|_w^2 . \tag{1}$$

Flow matching (v-prediction). For flow matching [43], we sample an isotropic Gaussian source point $p_i^{\text{src}} = \varepsilon_i$, with $\varepsilon_i \sim \mathcal{N}(\mathbf{0}, 0.2^2 \mathbf{I}_{3(T-1)})$ in normalized coordinates, and interpolate between source and target as $p_{i,t} = (1-t)p_i^{\text{src}} + t p_i$ for diffusion time $t \in [0, 1]$. The target velocity is $v_i^\star = (p_i - p_{i,t})/(1-t) = p_i - p_i^{\text{src}}$. Let $P_t \in \mathbb{R}^{N_p \times (T-1) \times 3}$ be the reshaped tensor corresponding to $\{p_{i,t}\}_{i=1}^{N_p}$. We train the model to predict $v_i^\star$:

$$\mathcal{L}_{\text{FM}} = \frac{1}{N_p} \sum_{i=1}^{N_p} \left\| g_{\theta,i}(P_t, P^{\text{obs}}, A, I, t) - v_i^\star \right\|_w^2 . \tag{2}$$

JiT-style (x-prediction). Flow matching and diffusion models for image generation are often applied in learned autoencoder latent spaces, enabling generation in a more compact and smooth

representation [44, 45]. In contrast, Just image Transformers (JiTs) [46] perform generation directly in pixel space. JiT argues that natural images lie on a low-dimensional manifold within pixel space, and therefore trains the model to directly predict the denoised image x_0.

Similarly, PointZero does not use a low-dimensional VAE to encode 3D flow, and we hypothesize that the data also lies on a low-dimensional manifold. Accordingly, we follow JiT and train the model to perform x-prediction: g_θ outputs the trajectory $\hat{p}_i = g_\theta(P_t, P^{\text{obs}}, A, I, t)_i$. Following the JiT training objective, we supervise the implied velocity $\hat{v}_i = (\hat{p}_i - p_{i,t})/(1-t)$ against the target velocity $v_i^\star$:

$$\mathcal{L}_{\text{JiT}} = \frac{1}{N_p} \sum_{i=1}^{N_p} \left\| \frac{g_{\theta,i}(P_t, P^{\text{obs}}, A, I, t) - p_{i,t}}{1-t} - v_i^\star \right\|_w^2 . \tag{3}$$

Following JiT, we clip the denominator $1-t$ to a minimum of 0.05 when computing both target and predicted velocities during training.

4.2 Dataset

A major bottleneck for scaling 3D dynamics learning is the absence of standardized datasets that capture action-conditioned, physically grounded interactions across diverse object types. Unlike video prediction, 3D dynamics learning requires temporally consistent geometry, explicit object states, and controllable interventions, making data collection particularly costly and fragmented across prior work. To this end, we contribute a new synthetic dataset spanning deformable, articulated, and rigid objects with motions driven by randomized interactions. Details regarding our dataset generation are provided in Appendix A.3.

4.3 Post-training Recipes

We study the utility of 3D point track completion as a pre-training objective for downstream tasks.

Post-training on end-effector state. For accurate scene-specific dynamics, we replace the conditioning on sparse partial point tracks with conditioning on robot end-effector (EEF) pose. Concretely, we swap the action tokens X_A for EEF tokens that encode the 6-DoF pose and gripper state at each timestep, embedded by a new MLP E_{EEF}. We initialize compatible components from the pre-trained model and fine-tune on the target scene.

Imitation-learning post-training. For policy learning, we remove partial point track conditioning and add an action head π_ϕ that predicts an EEF trajectory. With point-track supervision, we jointly predict EEF and point trajectories. See Section 5.7 and Appendices A.4 and A.9 for task-specific recipes.

5 Results

5.1 Implementation Details

Normalization and augmentations. We normalize each point cloud using the initial-frame centroid and 99th-percentile radius, and invert this transformation before evaluating errors in physical units. We then apply random rotations and Gaussian noise to both the point clouds and partial point track conditioning points to enhance robustness to noisy real-world measurements. We also apply visual augmentations to image observations, including random changes in brightness and contrast, salt-and-pepper noise, and Gaussian noise.

Training. We train PointZero on a node of 8$\times$ H100 GPUs for approximately 2 days per variant. Following JiT [46], we use logit-normal time sampling and additionally set t=0 with probability $p_{t=0} = 0.2$. At test time, both flow matching and JiT use four forward-Euler updates on a deterministic logit-normal quantile time grid with parameters $\mu = -3$ and $\sigma = 1$. We adopt a ViT-Base backbone (768 token dimension, 12 heads, 12 layers) with a 3-layer Perceiver-IO encoder and 4 query tokens in the pre-training checkpoints. Post-training omits DINO visual conditioning for efficiency.

Table 1: Dynamics prediction errors on held-out samples of our generated synthetic dataset. All baselines were trained on the proposed dataset. MDE, CD, and EMD are in centimeters; MSE is in 10^{-2} m^2. **Bold** marks the best result in each column, and underlining indicates the second-best result.

Method	Deformable				Articulated				Rigid			
	MDE ↓	MSE ↓	CD ↓	EMD ↓	MDE ↓	MSE ↓	CD ↓	EMD ↓	MDE ↓	MSE ↓	CD ↓	EMD ↓
GBND [47]	17.38	6.15	7.45	16.47	11.15	6.46	7.38	11.11	155.72	759.93	93.30	154.44
ParticleFormer [3]	17.35	6.15	7.42	16.44	10.86	6.46	7.11	10.83	155.04	760.43	92.76	153.77
PGND [1]	9.01	1.36	4.10	7.86	8.70	2.13	5.40	8.29	67.58	123.79	33.56	64.02
PTv3 [49]	9.20	1.59	3.91	8.28	8.25	2.31	5.25	8.00	61.97	114.77	29.87	58.57
PointZero-FM-mean-10	3.59	0.25	1.55	3.18	2.40	0.36	1.77	2.34	27.26	34.25	14.17	25.41
PointZero-FM-oracle-10	**2.80**	**0.15**	**1.24**	**2.41**	1.96	0.25	1.48	1.91	19.91	18.80	10.39	18.43
PointZero-Regression	4.08	0.32	1.74	3.63	3.85	0.66	2.57	3.68	32.14	44.58	16.43	29.78
PointZero-JiT-mean-10	3.62	0.26	1.56	3.22	2.36	0.37	1.70	2.30	28.06	39.25	14.65	26.35
PointZero-JiT-oracle-10	2.85	0.16	1.26	2.47	**1.95**	**0.24**	**1.44**	**1.89**	**19.46**	**17.89**	**10.30**	**17.99**

5.2 Baselines

We implement several strong existing learning-based dynamics methods, all *trained on our training dataset* with identical augmentations and a prediction horizon of $H = 10$. All methods use the same inputs, conditioning signals, output targets, and supervision; in particular, each baseline is adapted to condition on the same partial point tracks as PointZero.

Graph-Based Neural Dynamics (GBND) [47]: GBND downsamples the point cloud to 100 particles per scene, connects edges based on top-k nearest neighbors ($k = 5$) to form a spatial graph, and applies a GNN to predict next-step per-particle velocities. GBND is rolled out sequentially for longer predictions. **ParticleFormer [3]:** ParticleFormer uses the same input as GBND but uses a transformer for particle encoding and predicted velocity decoding. **Particle-Grid Neural Dynamics (PGND) [1]:** PGND uses PointNet [48] and a grid-based representation to predict next-step particle velocities. **Point Transformer v3 (PTv3) [49]:** PTv3 predicts next-step particle motions and performs sequential prediction in a similar way to PGND, but with a transformer architecture.

5.3 Evaluation Metrics

Following prior work [47, 3, 1], we evaluate prediction accuracy using the final-timestep point clouds. Given ground-truth points G and predicted points P, we report the mean squared error (MSE), mean distance error (MDE), the bi-directional Chamfer Distance (CD), and the Earth Mover's Distance (EMD). Formal definitions of these metrics are provided in Appendix A.2.

5.4 Synthetic Evaluation

We evaluate on held-out samples of our training dataset: approximately 32k scenes across all object categories. We also provide ablations over model and data size in Appendix A.5.

Comparison against baselines. Table 1 shows that every PointZero variant outperforms every baseline on every metric and object category. This suggests that a flexible but expressive transformer, generating entire motion sequences, is highly performant for learning diverse dynamics.

Training objectives. Generative models are capable of modeling arbitrary distributions given incomplete observations. We generate ten seeded predictions and report the lowest error independently for each scene and metric. This is a ground-truth oracle. The generative objectives, FM and JiT, outperform the regression variant, suggesting that they capture a multimodal distribution over plausible point trajectories. More comprehensive statistics and analysis are available in Appendix A.6.

In our experiments, JiT does not consistently outperform flow matching (PointZero-FM) when both models are trained for sufficiently long.

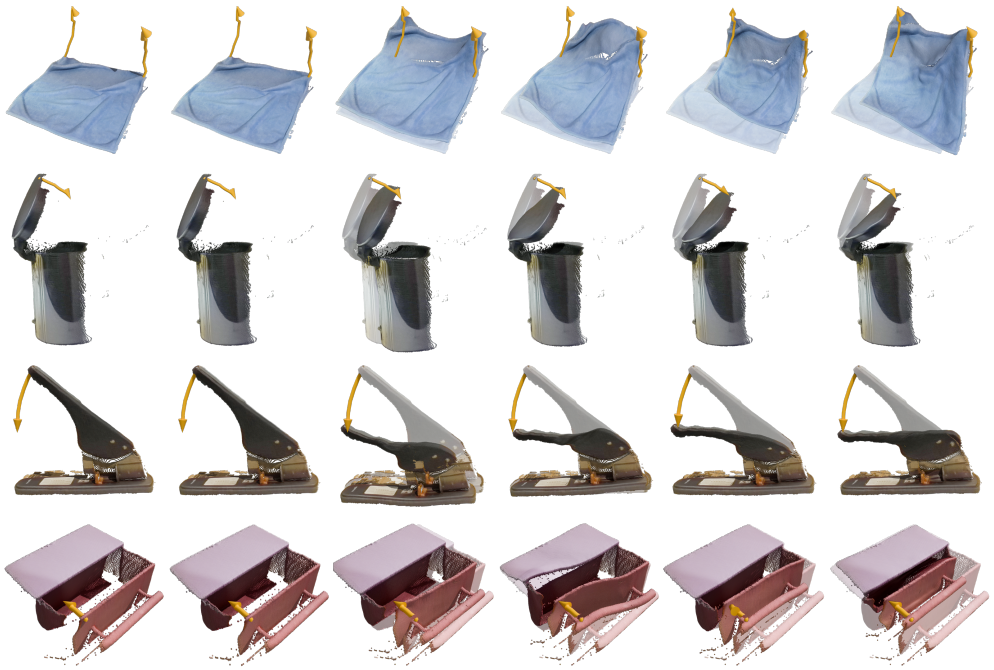

Figure 4: Real-world qualitative 3D point track completion evaluation. All methods are trained with identical inputs and conditioning, and with identical data. We find GBND often predicts zero motion, while PTv3 and PGND struggle to recover rigid (articulated) motions.

5.5 Real-World Zero-shot Partial Point Track Completion

We evaluate zero-shot generalization to real-world objects: methods are trained only on our simulation dataset and tested on novel real-world scenes without fine-tuning.

Data collection. We collect 124 interactions across 14 objects: 6 articulated, 5 deformable, and 3 rigid objects, where a human operator manipulates each object. We obtain action and point-track trajectory labels using FoundationStereo [25] and CoTracker3 [8]. To obtain action labels, we manually annotate one human-object contact point per human hand in the first frame and forward-propagate the trajectories.

Analysis. We display qualitative results in Fig. 4. Compared to baselines, PointZero excels at adhering to the input trajectory actions (orange arrows) it is conditioned on. PointZero point track completions are also more faithful to object-specific physical structure, with better preservation of cloth surface area and less deformation of rigid bodies. Table 9 shows that both FM and JiT outperform all baselines on 11 of 12 real-world metrics; PTv3 wins on rigid-object MSE. This suggests that for learning diverse dynamics, simple transformer architectures beat networks with inductive biases such as spatial grids [1].

5.6 Post-training: Robot-Conditioned 3D Dynamics Prediction

3D point track completion is an incomplete approximation of contact-rich real-world dynamics, as is common in robot manipulation. We post-train PointZero (Section 4.3) to predict 3D dynamics conditioned on robot end-effector state, and study the performance of its pre-trained spatial prior. We evaluate PointZero on the six-scene benchmark from PGND [1] and compare against strong *application-specific* baselines. For each scene, we post-train a separate instance of PointZero using the same training set used to train the PGND baseline. We post-train with the JiT objective and LoRA. We also train PointZero-Scratch with the same objective, random initialization, and all parameters trainable. The results suggest two things: (1) PointZero convincingly outperforms state-of-the-art baselines on 4/6 scenes, and (2) pre-training is critical to its strong predictions.

Table 2: Dynamics prediction errors on the PGND benchmark [1] (centimeters). The PGND method is trained only on scene-specific interactions. We evaluate PointZero both trained from scratch on the scene-specific interactions (PointZero-Scratch), and also fine-tuned (PointZero-FT) on the scene-specific interactions from the pre-trained checkpoint. Both PointZero rows report a per-scene, per-metric best-of-10 oracle (Section 5.6). **Bold** marks the best result in each column.

Method	Bread			Paperbag			Cloth			Box			Rope			Sloth		
	MDE↓	CD↓	EMD↓	MDE↓	CD↓	EMD↓	MDE↓	CD↓	EMD↓	MDE↓	CD↓	EMD↓	MDE↓	CD↓	EMD↓	MDE↓	CD↓	EMD↓
GBND [17]	3.1	3.1	1.6	3.0	4.2	1.6	7.7	8.3	3.5	4.5	6.2	3.2	6.2	7.3	3.6	7.8	6.4	3.2
PGND [1]	2.0	1.8	1.0	**1.6**	2.1	0.9	4.5	4.3	2.2	**2.2**	**1.5**	1.6	3.9	3.8	2.1	4.3	3.3	1.7
PointZero-Scratch	4.2	3.9	1.9	4.2	4.4	1.9	5.6	4.7	2.1	3.7	3.9	2.1	9.3	10.8	5.0	7.3	5.9	2.8
PointZero-FT	**1.5**	**1.1**	**0.6**	**1.6**	**2.0**	**0.8**	**4.0**	**3.4**	**1.5**	2.4	2.7	**1.4**	**3.3**	**3.3**	**1.6**	**3.9**	**3.0**	**1.5**

Table 3: Imitation learning success (%). PointZero uses 20 labeled demos and 100 actionless videos per task.

Task	3PoinTr	DP3	DP	ATM	PointZero
Simulation					
Blockstack	90.9	44.4	45.4	40.0	**99.8**
Microwave	80.8	31.4	32.0	30.4	**93.1**
Glass	95.2	74.9	57.5	3.9	**95.9**
Real-world					
Drawer	90.0	70.0	–	30.0	**100.0**
Cup	**100.0**	80.0	–	30.0	**100.0**
Paper	**90.0**	10.0	–	0.0	70.0
Sock	**90.0**	20.0	–	0.0	**90.0**

Table 4: Pre-training **with** downstream track supervision. Success (%); 20 demos, no extra videos.

Recipe	Blockstack	Microwave	Glass	Avg.
Scratch	**93.5**	81.4	66.6	80.5
Pretrained	93.2	**91.9**	**79.5**	**88.2**

Table 5: Pre-training **without** downstream track supervision. Success (%); 20 demos, no extra videos.

Recipe	Blockstack	Microwave	Glass	Avg.
Scratch	83.8	89.5	49.1	74.1
Pretrained	**85.2**	**90.1**	**64.6**	**80.0**

We additionally evaluate PointZero in a zero-shot setting on the PGND benchmark. It achieves competitive performance with scene-specific baselines, *despite never having interacted with the objects.* We provide these results and additional details in Appendix A.8.

5.7 Post-training: Imitation Learning for Robot Manipulation

We test whether the trained representation effectively adapts to robot manipulation using the experimental evaluation protocol from 3PoinTr [2]. We independently post-train PointZero to predict robot actions in three simulation tasks and four real-world tasks. For each task, we use 20 expert demonstrations with robot action labels, and 100 additional videos of expert demonstrations without any action labels. We provide details about the tasks and policy-training setup in Appendix A.9.

Baselines. **DP3** [50] and **Diffusion Policy** [51] are performant behavior cloning methods that encode observations and then use conditional diffusion models to generate action chunks. **3PoinTr** [2] and **ATM** [40] pre-train point-track prediction models from videos, and then condition policies on the point-track predictions.

Analysis. PointZero achieves the highest or joint-highest success rate on six of seven manipulation tasks (Table 3). Using only 20 action-labeled demonstrations per simulation task and no additional actionless videos, pre-training increases average success from 80.5% to 88.2% with auxiliary point-track supervision (Table 4); without it, the pre-trained recipe achieves 80.0% versus 74.1% from scratch (Table 5). The latter recipe freezes the pre-trained point-processing stream, whereas Scratch trains the full model (Appendix A.4). Both comparisons use 1,000 rollouts per simulation task; bold entries mark the better variant. The results suggest that 3D point track completion as a pre-training objective helps downstream imitation learning applications. In particular, the isolated pre-training experiments in Table 4 and Table 5 suggest 3D point-track completion directly improves performance.

6 Conclusion

We present PointZero, a 3D dynamics model trained to do *3D point track completion*: given a single RGB-D observation and a small number of sparse point trajectories, the model predicts dense future 3D tracks for all observed points. This objective provides a scalable way to learn dynamics priors without requiring embodiment-specific annotations, using a metric point representation that can represent deformable, articulated, and rigid objects.

Across synthetic and real-world evaluations, PointZero substantially improves dense point-track completion over adapted 3D dynamics baselines. More importantly, the learned representation transfers effectively to downstream tasks: after post-training, PointZero improves scene-specific action-conditioned dynamics prediction and achieves strong performance on robot manipulation tasks. These results suggest that 3D point track completion can serve as a useful bridge between scalable robot-free dynamics pre-training and downstream embodied intelligence applications.

Limitations and Future Work. PointZero remains limited by the coverage and realism of its pre-training data. Although our synthetic dataset spans several object classes, it does not capture the full diversity of real-world materials, contact-rich hand-object interactions, cluttered scenes, or long-horizon dynamics. Future work should augment simulation with pseudo-annotations from real-world videos using 3D reconstruction and point tracking methods. PointZero also uses relatively simple conditioning signals: partial point tracks during pre-training and end-effector pose during action-conditioned post-training. Richer interaction representations, such as contact locations and forces, may further improve accuracy. Finally, our downstream evaluations remain task-specific and relatively small-scale; future work should study multi-scene and multi-task transfer across a broader range of applications.

Acknowledgments and Disclosure of Funding

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant Nos. DGE2140739 and DGE2631988. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

This work used Bridges-2 at Pittsburgh Supercomputing Center through allocation CIS250005p.

Competing interests. The authors declare no competing interests.

References

[1] Kaifeng Zhang, Baoyu Li, Kris Hauser, and Yunzhu Li. Particle-grid neural dynamics for learning deformable object models from rgb-d videos. In *Proceedings of Robotics: Science and Systems (RSS)*, 2025.

[2] Adam Hung, Bardienus Pieter Duisterhof, and Jeffrey Ichnowski. 3pointr: 3d point tracks for robot manipulation pretraining from casual videos. *arXiv preprint arXiv:2603.08485*, 2026.

[3] Suning Huang, Qianzhong Chen, Xiaohan Zhang, Jiankai Sun, and Mac Schwager. Particleformer: A 3d point cloud world model for multi-object, multi-material robotic manipulation. *arXiv preprint arXiv:2506.23126*, 2025.

[4] Ruijie Zheng, Dantong Niu, Yuqi Xie, Jing Wang, Mengda Xu, Yunfan Jiang, Fernando Castañeda, Fengyuan Hu, You Liang Tan, Letian Fu, Trevor Darrell, Furong Huang, Yuke Zhu, Danfei Xu, and Linxi Fan. Egoscale: Scaling dexterous manipulation with diverse egocentric human data, 2026. arXiv:2602.16710.

[5] Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation, 2023. arXiv:2312.13139.

[6] Wenlong Huang, Yu-Wei Chao, Arsalan Mousavian, Ming-Yu Liu, Dieter Fox, Kaichun Mo, and Li Fei-Fei. Pointworld: Scaling 3d world models for in-the-wild robotic manipulation, 2026. arXiv:2601.03782.

[7] Jay Karhade, Nikhil Keetha, Yuchen Zhang, Tanisha Gupta, Akash Sharma, Sebastian Scherer, and Deva Ramanan. Any4d: Unified feed-forward metric 4d reconstruction. *arXiv preprint arXiv:2512.10935*, 2025.

[8] Nikita Karaev, Iurii Makarov, Jianyuan Wang, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. CoTracker3: Simpler and better point tracking by pseudo-labelling real videos. 2024.

[9] Tiantian Liu, Adam W. Bargteil, James F. O'Brien, and Ladislav Kavan. Fast simulation of mass-spring systems. *ACM Trans. Graph.*, 32(6), November 2013.

[10] 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. *ICCV*, 2025.

[11] Marcos García, César Mendoza, Luis Pastor, and Angel Rodríguez. Optimized linear fem for modeling deformable objects. *Computer Animation and Virtual Worlds*, 17(3-4):393–402, 2006.

[12] Miles Macklin, Matthias Müller, Nuttapong Chentanez, and Tae-Yong Kim. Unified particle physics for real-time applications. *ACM Trans. Graph.*, 33(4), July 2014.

[13] Deborah Sulsky, Shi-Jian Zhou, and Howard L. Schreyer. Application of a particle-in-cell method to solid mechanics. *Computer Physics Communications*, 87(1):236–252, 1995. Particle Simulation Methods.

[14] 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. *arXiv preprint arXiv:2303.05512*, 2023.

[15] 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*, 2019.

[16] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International conference on machine learning*, pages 8459–8468. PMLR, 2020.

[17] Kaifeng Zhang, Baoyu Li, Kris Hauser, and Yunzhu Li. Adaptigraph: Material-adaptive graph-based neural dynamics for robotic manipulation. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

[18] William F Whitney, Jacob Varley, Deepali Jain, Krzysztof Choromanski, Sumeet Singh, and Vikas Sindhwani. Modeling the real world with high-density visual particle dynamics. *arXiv preprint arXiv:2406.19800*, 2024.

[19] Tongxuan Tian, Haoyang Li, Bo Ai, Xiaodi Yuan, Zhiao Huang, and Hao Su. Diffusion dynamics models with generative state estimation for cloth manipulation. *Conference on Robot Learning (CoRL)*, 2025.

[20] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *CVPR*, 2024.

[21] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

[22] Nikhil Keetha, Norman Müller, Johannes Schönberger, Lorenzo Porzi, Yuchen Zhang, Tobias Fischer, Arno Knapitsch, Duncan Zauss, Ethan Weber, Nelson Antunes, Jonathon Luiten, Manuel Lopez-Antequera, Samuel Rota Bulò, Christian Richardt, Deva Ramanan, Sebastian Scherer, and Peter Kontschieder. MapAnything: Universal feed-forward metric 3D reconstruction. In *International Conference on 3D Vision (3DV)*. IEEE, 2026.

[23] Bardienus Duisterhof, Lojze Zust, Philippe Weinzaepfel, Vincent Leroy, Yohann Cabon, and Jerome Revaud. Mast3r-sfm: a fully-integrated solution for unconstrained structure-from-motion, 2024. arXiv:2409.19152.

[24] Bowen Wen, Wei Yang, Jan Kautz, and Stan Birchfield. Foundationpose: Unified 6d pose estimation and tracking of novel objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17868–17879, 2024.

[25] Bowen Wen, Matthew Trepte, Joseph Aribido, Jan Kautz, Orazio Gallo, and Stan Birchfield. Foundationstereo: Zero-shot stereo matching. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5249–5260, 2025.

[26] Bardienus P Duisterhof, Jan Oberst, Bowen Wen, Stan Birchfield, Deva Ramanan, and Jeffrey Ichnowski. Rayst3r: Predicting novel depth maps for zero-shot object completion. *NeurIPS*, 2025.

[27] Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. DINO-WM: World models on pre-trained visual features enable zero-shot planning. In *International Conference on Machine Learning*, 2025.

[28] Boyuan Chen, Diego Marti Monso, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion, 2024. arXiv:2407.01392.

[29] Hongyu Li, Bowen Wen, Xinghao Zhu, Yixuan Wang, Yilun Du, Yunzhu Li, George Konidaris, Stan Birchfield, Soha Pouya, Chenran Li, and Yan Chang. Hydra-0: Action flow for generalist world modeling and control. *arXiv preprint arXiv:2608.18077*, 2026.

[30] Jianing Zhang, Chenhao Zheng, Yajun Yang, Max Argus, Rustin Soraki, Winson Han, Taira Anderson, Chun-Liang Li, Shuo Liu, Jiafei Duan, Zhongzheng Ren, Jieyu Zhang, and Ranjay Krishna. MolmoMotion: Forecasting point trajectories in 3D with language instruction. *arXiv preprint arXiv:2606.18558*, 2026.

[31] Jiahui Lei, Qianqian Wang, Trevor Darrell, and Angjoo Kanazawa. World motion models: Flexible sequence modeling of SE(3) trajectories. *arXiv preprint arXiv:2610.01742*, 2026.

[32] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollar, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022.

[33] Shengyi Qian, Kaichun Mo, Valts Blukis, David F Fouhey, Dieter Fox, and Ankit Goyal. 3d-mvp: 3d multiview pretraining for manipulation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22530–22539, 2025.

[34] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.

[35] Adam Hung, Bardienus P. Duisterhof, Deva Ramanan, and Jeffrey Ichnowski. Modality-autoregressive world-action models, 2026. arXiv:2609.17524.

[36] Ge Yan, Jinghao Liu, Yuzhi Fan, Lei Cai, Minwen Liao, Jesse Zhang, and Dieter Fox. Flex-π: A multi-stream world-action model with compute flexibility. *arXiv preprint arXiv:2608.10860*, 2026.

[37] Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World action models are zero-shot policies, 2026. arXiv:2602.15922.

[38] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3M: A universal visual representation for robot manipulation. In Karen Liu, Dana Kulic, and Jeff Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 892–909. PMLR, 14–18 Dec 2023.

[39] Kenneth Shaw, Shikhar Bahl, and Deepak Pathak. Videodex: Learning dexterity from internet videos, 2022. arXiv:2212.04498.

[40] Chuan Wen, Xingyu Lin, John Ian Reyes So, Kai Chen, Qi Dou, Yang Gao, and Pieter Abbeel. Any-point trajectory modeling for policy learning. In *Robotics: Science and Systems*, 2024.

[41] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver IO: A general architecture for structured inputs & outputs. In *ICLR*, 2022.

[42] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell

Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision. arXiv:2304.07193, 2023.

[43] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023. arXiv:2210.02747.

[44] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[45] Quan Dao, Hao Phung, Binh Nguyen, and Anh Tran. Flow matching in latent space. *arXiv preprint arXiv:2307.08698*, 2023.

[46] Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise, 2026. arXiv:2511.13720.

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

[48] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

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

[50] Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3D diffusion policy: Generalizable visuomotor policy learning via simple 3D representations. In *Robotics: Science and Systems*, 2024.

[51] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems*, 2023.

[52] Thomas Lips, Victor-Louis De Gusseme, and Francis wyffels. Learning keypoints for robotic cloth manipulation using synthetic data, 2024. arXiv:2401.01734.

[53] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Blender Institute, Amsterdam.

[54] Poly Haven. Poly haven: Free high-quality assets, 2026.

[55] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[56] Genesis Authors. Genesis: A generative and universal physics engine for robotics and beyond, December 2024.

[57] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, Thomas Kipf, Abhijit Kundu, Dmitry Lagun, Issam Laradji, Hsueh-Ti (Derek) Liu, Henning Meyer, Yishu Miao, Derek Nowrouzezahrai, Cengiz Oztireli, Etienne Pot, Noha Radwan, Daniel Rebain, Sara Sabour, Mehdi S. M. Sajjadi, Matan Sela, Vincent Sitzmann, Austin Stone, Deqing Sun, Suhani Vora, Ziyu Wang, Tianhao Wu, Kwang Moo Yi, Fangcheng Zhong, and Andrea Tagliasacchi. Kubric: a scalable dataset generator. 2022.

A Appendix

A.1 Training Loss Weighting

We give more weight to later timesteps to emphasize longer-horizon predictions, where uncertainty is greater. For future frames $\tau \in \{2, \ldots, T\}$, we use $\|u_i\|_w^2 = \frac{1}{3(T-1)} \sum_{\tau=2}^{T} [0.1 + 0.9((\tau - 1)/T)^2] \|u_{i,\tau}\|_2^2$.

A.2 Dynamics Prediction Evaluation Metrics

The definitions below apply to PointZero's corresponding, equal-cardinality final-frame point sets; $\Pi(P, G)$ denotes permutation matrices.

Mean Squared Error (MSE):

$$\mathcal{L}_{\text{MSE}}(P, G) = \frac{1}{N_p} \sum_{i=1}^{N_p} \|P_i - G_i\|_2^2. \tag{4}$$

Mean Distance Error (MDE):

$$\mathcal{L}_{\text{MDE}}(P, G) = \frac{1}{N_p} \sum_{i=1}^{N_p} \|P_i - G_i\|_2. \tag{5}$$

Bi-Directional Chamfer Distance (CD):

$$\mathcal{L}_{\text{CD}}(P, G) = \frac{1}{2N_p} \sum_{p \in P} \min_{g \in G} \|p - g\|_2 + \frac{1}{2N_p} \sum_{g \in G} \min_{p \in P} \|g - p\|_2. \tag{6}$$

Earth Mover's Distance (EMD):

$$\mathcal{L}_{\text{EMD}}(P, G) = \min_{\pi \in \Pi(P,G)} \frac{1}{N_p} \sum_{i=1}^{N_p} \sum_{j=1}^{N_p} \pi_{ij} \|P_i - G_j\|_2. \tag{7}$$

For the PGND benchmark, EMD is computed using more points than the other metrics, which can yield lower reported distances.

A.3 Dataset Generation Details

Our dataset comprises deformable, articulated, and rigid object interactions. For all datasets, we randomize camera location and intrinsics and use a single-view RGB-D observation. This models the partial visibility and various viewpoints encountered in the real world.

Deformables. We follow prior work [52] to achieve procedural cloth generation.

- **Mesh Generation**: We generate *towels*, *t-shirts* and *shorts* meshes. We randomize both the scale and the proportions.
- **Physics**: We use NVIDIA FleX [12] to model deformable physics. We apply one or two action trajectories to the cloth, modeling interactions such as *fold*, *lift*, *drop*, or *push*. Concretely, given constraints on location of action points, we solve for the location of all other points that would produce static equilibrium. To enhance motion diversity, we randomize physics parameters such as stiffness and drag.
- **Appearance**: We render the scenes with Blender [53], with random cloth and background textures from PolyHaven [54].

Articulated Objects We use the PartNet-Mobility dataset [55] to retrieve articulated objects, and place them in Genesis [56] for simulating motion. We randomly pick a revolute or prismatic joint to articulate, and set a random configuration in its viable range to produce static equilibrium. This alone may result in an invisible articulation, or very small motions. To address this issue, we reject samples where the 2D optical flow of the sequence is small using privileged information in simulation.

Rigid Objects Finally, we aim to instill an understanding of rigid body dynamics and collision into PointZero. We modify the Kubric engine [57] to include ground-truth 3D trajectories and fewer objects (up to 3 instead of 23 objects).

Sampling Action Trajectories For partial point trajectory completion training, we automate the extraction of partial point trajectories. For deformable objects, we save the action trajectories that were used as kinematic constraints in NVIDIA FleX [12] to be used as training input. For articulated and rigid bodies, we pick 3D point tracks from the k largest displacements.

Dataset Mixing. The recorded training weights are 0.33 each for shorts, T-shirts, and towels, 1.0 for articulated objects, and 0.33 for rigid objects, giving category probabilities of approximately 42.67%, 43.10%, and 14.22%, respectively. This gives an approximate category ratio of $3:3:1$, reflecting our emphasis on learning more complex non-rigid and articulated dynamics while still exposing the model to rigid-body motion and collisions.

A.4 Additional Manipulation Comparisons

Tables 4 and 5 report the pre-training ablations using only 20 demonstrations per simulation task, with no additional actionless videos. In Table 4, PointZero initializes from JiT pre-training and scratch initializes randomly; both train all parameters with point-trajectory regression and an $\ell_{2,1}$ action loss, using a separate action branch with learned queries. In Table 5, both use only the action loss; PointZero keeps its pre-trained point-processing stream fixed while training the remaining parameters, whereas scratch trains the full model. With downstream track supervision, pre-training improves two tasks and the average (80.5% to 88.2%), while Blockstack changes from 93.5% to 93.2%. Under this protocol, Diffusion Policy [51] and DP3 [50] achieve average success rates of 45.0% and 50.2%, respectively; their per-task results are listed in Table 3. Without downstream point-track supervision, pre-trained PointZero outperforms its scratch variant on all three tasks, with the largest gap on Glass (64.6% vs. 49.1%). DP3 outperforms PointZero without downstream track supervision on Glass (74.9% vs. 64.6%).

A.5 Ablations

Table 6: Data-scaling ablation on the articulated split of the real-world dataset: performance degrades gracefully as the pre-training dataset is reduced. Error values are in centimeters; MSE is in cm^2. Training epochs are scaled inversely with the data fraction (250 epochs at 10%, 500 at 5%). **Bold** marks the best result for each metric.

Data Fraction	MDE↓	MSE↓	CD↓	EMD↓
100%	**2.2**	**12.1**	**1.4**	**2.0**
10%	3.1	49.6	2.0	2.9
5%	3.3	64.3	2.0	3.1

Table 7: Model-size ablation on the articulated split of the real-world dataset. Small uses 384 token dimensions, 6 heads, and 6 layers (45M parameters) versus the Base configuration (768 dimensions, 12 heads, 12 layers). Error values are in centimeters; MSE is in cm^2. **Bold** marks the best result for each metric.

Size	MDE↓	MSE↓	CD↓	EMD↓
Base	**2.2**	**12.1**	**1.4**	**2.0**
Small	3.1	37.4	1.8	3.0

We study the impact of the training data fraction and of model size.

Training data fraction. Table 6 reports performance on the articulated split of the real-world set as the pre-training dataset is reduced, with training epochs scaled inversely to hold the optimization budget comparable. Performance degrades gracefully: reducing the data to 10% increases MDE from 2.2 to 3.1 cm, and further reduction to 5% yields only a small additional drop. This suggests the pre-training objective extracts useful dynamics priors even from a fraction of the dataset, while the full dataset remains clearly beneficial.

Model size. We additionally train a Small variant from scratch (384 token dimensions, 6 heads, 6 layers) with otherwise identical inputs and training data. Table 7 reports performance in the same setting: shrinking the model degrades all metrics substantially, suggesting that further scaling the Base model could yield additional gains.

A.6 Additional Synthetic Data Point Dynamics Completion Results

In Table 8, we expand Table 1 with additional evaluations of PointZero-JiT and PointZero-FM. For each sample, we generate ten predictions using different random noise initializations. The "-oracle-10" rows select the lowest error separately for each scene and metric; "-first" uses the first prediction, and "-mean-10" averages scalar errors over all ten predictions. For "-medoid-10", each corresponding point's final position is selected from its ten sampled positions to minimize the summed Euclidean distance to the other nine, without using ground truth. Different points may come from different samples. The "-pooled-scalar-median-10" rows take each metric's median over scene–prediction pairs within each dataset, then average dataset medians. These distribution summaries are not directly comparable to mean-error rows.

PointZero-JiT-first outperforms PointZero-Regression on 11 of 12 metrics; the exception is rigid-object MSE (48.627 versus 44.576 in $10^{-2}\,\mathrm{m}^2$). Both per-metric best-of-10 oracles outperform regression on all 12 metrics, showing that sampling can produce candidates with lower reconstruction error.

Table 8: Dynamics prediction errors on held-out samples of our generated synthetic dataset. All baselines were trained on the proposed dataset. MDE, CD, and EMD are in centimeters; MSE is in $10^{-2}\,\mathrm{m}^2$. First-sample, mean, pointwise medoid, per-metric oracle, and pooled scalar median statistics use ten seeded predictions and are defined in Appendix A.6. Pooled medians summarize a different statistic from mean errors; no cross-statistic ranking is applied.

Method	Deformable				Articulated				Rigid			
	MDE ↓	MSE ↓	CD ↓	EMD ↓	MDE ↓	MSE ↓	CD ↓	EMD ↓	MDE ↓	MSE ↓	CD ↓	EMD ↓
GBND [47]	17.38	6.15	7.45	16.47	11.15	6.46	7.38	11.11	155.72	759.93	93.30	154.44
ParticleFormer [3]	17.35	6.15	7.42	16.44	10.86	6.46	7.11	10.83	155.04	760.43	92.76	153.77
PGND [1]	9.01	1.36	4.10	7.86	8.70	2.13	5.40	8.29	67.58	123.79	33.56	64.02
PTv3 [49]	9.20	1.59	3.91	8.28	8.25	2.31	5.25	8.00	61.97	114.77	29.87	58.57
PointZero-Regression	4.08	0.32	1.74	3.63	3.85	0.66	2.57	3.68	32.14	44.58	16.43	29.78
PointZero-FM-first	3.60	0.25	1.56	3.17	2.39	0.35	1.77	2.33	28.97	38.79	15.16	27.20
PointZero-FM-mean-10	3.59	0.25	1.55	3.18	2.40	0.36	1.77	2.34	27.26	34.25	14.17	25.41
PointZero-FM-pooled-scalar-median-10	3.21	0.14	1.30	2.80	1.75	0.05	1.47	1.74	11.65	1.85	7.03	11.27
PointZero-FM-medoid-10	3.26	0.21	1.40	2.87	2.23	0.33	1.65	2.18	24.46	28.57	12.59	22.74
PointZero-FM-oracle-10	2.80	0.15	1.24	2.41	1.96	0.25	1.48	1.91	19.91	18.80	10.39	18.43
PointZero-JiT-first	3.65	0.26	1.59	3.24	2.36	0.36	1.70	2.29	30.26	48.63	16.03	28.67
PointZero-JiT-mean-10	3.62	0.26	1.56	3.22	2.36	0.37	1.70	2.30	28.06	39.25	14.65	26.35
PointZero-JiT-pooled-scalar-median-10	3.27	0.15	1.31	2.87	1.65	0.06	1.37	1.64	11.23	1.86	6.96	11.01
PointZero-JiT-medoid-10	3.32	0.22	1.42	2.92	2.23	0.33	1.61	2.17	24.93	33.08	12.81	23.32
PointZero-JiT-oracle-10	2.85	0.16	1.26	2.47	1.95	0.24	1.44	1.89	19.46	17.89	10.30	17.99

A.7 Real-World Dynamics Prediction Quantitative Results

In Table 9, we report point track completion error metrics for PointZero and baselines on our real-world dataset. All methods are trained on the same synthetic dataset and evaluated zero-shot on the unseen real-world objects. Both PointZero-FM and PointZero-JiT outperform all baselines on

Table 9: Dynamics prediction errors on the real-world dataset (centimeters; MSE in cm^2). All baselines were trained on the proposed simulation dataset. "-first" rows use a single sample; PointZero-FM and PointZero-JiT report all metrics of the sample with the lowest MDE out of 10 (best-of-10). **Bold** marks the best result in each column, and underlining indicates the second-best result.

Method	Deformable				Articulated				Rigid			
	MDE ↓	MSE ↓	CD ↓	EMD ↓	MDE ↓	MSE ↓	CD ↓	EMD ↓	MDE ↓	MSE ↓	CD ↓	EMD ↓
GBND	6.179	81.114	4.039	6.069	3.051	24.148	2.089	2.912	4.379	34.418	2.085	4.307
ParticleFormer	6.156	80.928	4.022	6.044	3.026	24.009	2.078	2.887	4.367	34.332	2.081	4.296
PGND	3.757	27.054	2.244	3.646	2.828	20.137	1.702	2.666	2.650	13.210	1.212	2.546
PTv3	4.705	35.109	2.975	4.606	4.430	43.665	2.645	4.284	2.487	**11.604**	1.108	2.371
PointZero-FM-first	3.739	28.765	2.293	3.642	2.046	10.152	1.168	1.883	2.711	15.697	1.216	2.542
PointZero-FM	2.923	**17.385**	1.852	**2.827**	**1.720**	**8.096**	**1.031**	**1.580**	**2.322**	12.484	**1.078**	**2.182**
PointZero-Regression	3.643	25.258	2.416	3.584	2.192	11.227	1.322	2.060	2.654	14.647	1.133	2.558
PointZero-JiT-first	3.706	32.545	2.294	3.636	2.317	12.608	1.338	2.166	2.815	15.969	1.276	2.656
PointZero-JiT	**2.915**	19.162	**1.837**	2.846	1.914	9.670	1.156	1.767	2.398	12.376	1.103	2.238

Table 10: Evaluation of different 3D dynamics architectures on zero-shot predictions on the PGND benchmark [1]. All methods were trained on synthetic data from the PointZero dataset, and evaluated on unseen PGND scenes. Errors are in centimeters. **Bold** marks the best result in each column.

Method	Box			Paperbag			Cloth			Rope			Sloth			Bread		
	MDE↓	CD↓	EMD↓	MDE↓	CD↓	EMD↓	MDE↓	CD↓	EMD↓	MDE↓	CD↓	EMD↓	MDE↓	CD↓	EMD↓	MDE↓	CD↓	EMD↓
PGND-ZS	3.1	2.0	3.0	2.9	1.7	2.7	5.9	3.0	5.6	8.6	5.1	8.5	8.3	3.1	7.5	2.9	1.3	2.8
GBND-ZS	2.5	1.6	2.4	3.7	2.2	3.6	8.9	4.5	8.7	13.0	8.8	12.9	11.1	4.8	10.3	3.3	1.6	3.3
ParticleFormer-ZS	2.5	1.6	2.4	3.7	2.1	3.6	8.9	4.5	8.7	13.0	8.8	12.9	11.1	4.8	10.3	3.3	1.6	3.3
PTv3-ZS	3.4	2.0	3.3	3.9	2.2	3.7	7.2	3.6	6.9	7.7	4.9	7.6	6.3	2.5	5.2	2.6	1.2	2.5
PointZero-FM-ZS	**2.2**	**1.4**	**2.1**	**2.7**	**1.6**	**2.5**	5.1	2.5	4.9	**5.4**	**3.5**	**5.3**	**5.6**	**2.3**	**4.9**	**1.9**	**0.9**	**1.8**
PointZero-Regression-ZS	2.5	1.6	2.3	2.8	1.7	2.7	4.9	2.5	4.7	5.9	3.6	5.8	6.3	2.5	5.5	2.4	1.1	2.4
PointZero-JiT-ZS	2.3	1.5	**2.1**	**2.7**	**1.6**	**2.5**	**4.7**	**2.4**	**4.4**	5.8	3.8	5.8	6.1	2.6	5.5	2.1	1.0	2.0

11 of 12 metrics; PTv3 achieves the lowest rigid-object MSE (11.604 cm^2, versus 12.376 cm^2 for PointZero-JiT).

A.8 PGND Zero-shot Evaluation

We train PointZero and several 3D dynamics architecture baselines on point track completion on the PointZero synthetic dataset. Then, we evaluate the zero-shot performance on the PGND benchmark. To enable zero-shot predictions without fine-tuning the models to condition on robot pose, we extract a single object-point trajectory per PGND sample, and condition on this track.

Results are shown in Table 10. PointZero-FM-ZS achieves an average MDE of approximately 3.8 cm across the six scenes, compared with 5.2 cm for the strongest baseline, PTv3-ZS, a reduction of approximately 26%.

A.9 Imitation Learning Experimental Setup

Following 3PoinTr [2], we evaluate on three simulation tasks: (1) stacking blocks, (2) opening a microwave, and (3) righting a fallen glass. We also evaluate on four real-world tasks: (1) opening a drawer, (2) righting a fallen glass, (3) picking up a crumpled piece of paper and placing it in the trash, and (4) folding a sock in half. The position and orientation of the objects are varied across trials, and the same initial configurations are used across all methods. Figure 5 shows images of all the tasks.

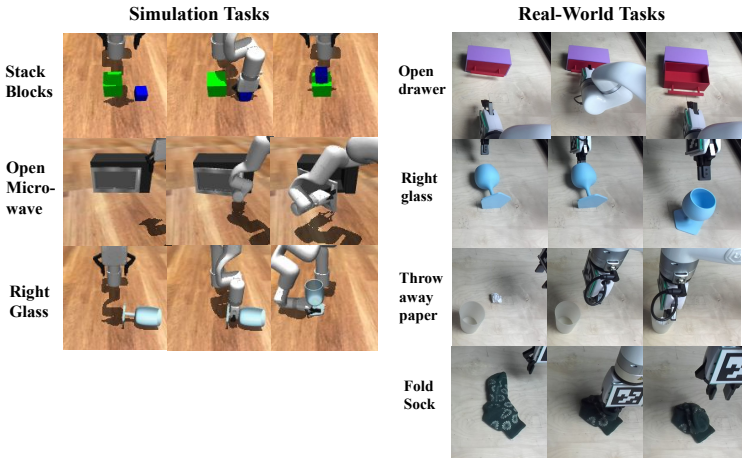

Figure 5: Visualizations of the simulation and real-world tasks used for imitation learning evaluation.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Yes: we precisely highlight the contribution of our model without overclaiming; claims align with the Results section and Appendix.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Yes: we discuss limitations in the Conclusion ("Limitations and Future Work"): synthetic pre-training coverage, simple conditioning, and scale of downstream evaluations.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.

- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: We do not provide any theorems or proofs in this paper.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Methods describe architecture and objectives; Results covers preprocessing, augmentations, training settings, baselines, and metrics (definitions in Appendix A.2); Appendix A.3 documents dataset construction.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case

of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We plan to release the code, model checkpoints, and dataset with instructions to support reproduction and adoption. .

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Results (*Implementation Details*) specifies preprocessing, augmentations, compute, sampling and training choices, integration steps, backbone sizes, and aligned baseline training; downstream sections specify PGND and imitation-learning protocols; the appendix adds dataset and evaluation detail.

   Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We run the generative methods across 10 random seeds and report mean, median, and best metrics. This supports a more comprehensive evaluation of the distribution of outcomes for these methods.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Results reports pre-training on 8$\times$ H100 GPUs for approximately two days per variant (fine-tuning compute is smaller and not itemized separately).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: The work uses simulation and standard robotics benchmarks with citations; we are not aware of ethics violations for this submission.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: PointZero contributes to research on 3D dynamics models for robotics. Positive impacts include lowering the data burden for deploying robots to new tasks (since pre-training does not require robot action labels) and enabling better simulation and AR/VR applications. As with manipulation research broadly, improved policies carry dual-use risks, including potential applications in surveillance or labor-displacing automation; we note these are general to the field rather than specific to our contribution. Our released artifacts (synthetic dataset, dynamics model checkpoints) are foundational rather than tied to a specific deployed system, and we are not aware of capabilities unlocked beyond those of existing 3D dynamics or manipulation models. Due to space constraints, we do not include a dedicated section in the main paper.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: Our release targets synthetic dynamics data and a 3D dynamics model for robotics; it is not in the high-risk category described in the question.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Licenses/assets acknowledged for the generated data.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The synthetic dataset and real-world evaluation data are described in the main text, Appendix A.3, and Section 5.5; we intend to ship documentation with the released assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: There are no crowdsourcing studies or formal human-subjects protocols with recruited participants.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not report IRB-reviewed human-subjects research.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs are not used in any non-standard way.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.