Dex4D: Task-Agnostic Point Track Policy for Sim-to-Real Dexterous Manipulation

Author Names Omitted for Anonymous Review.

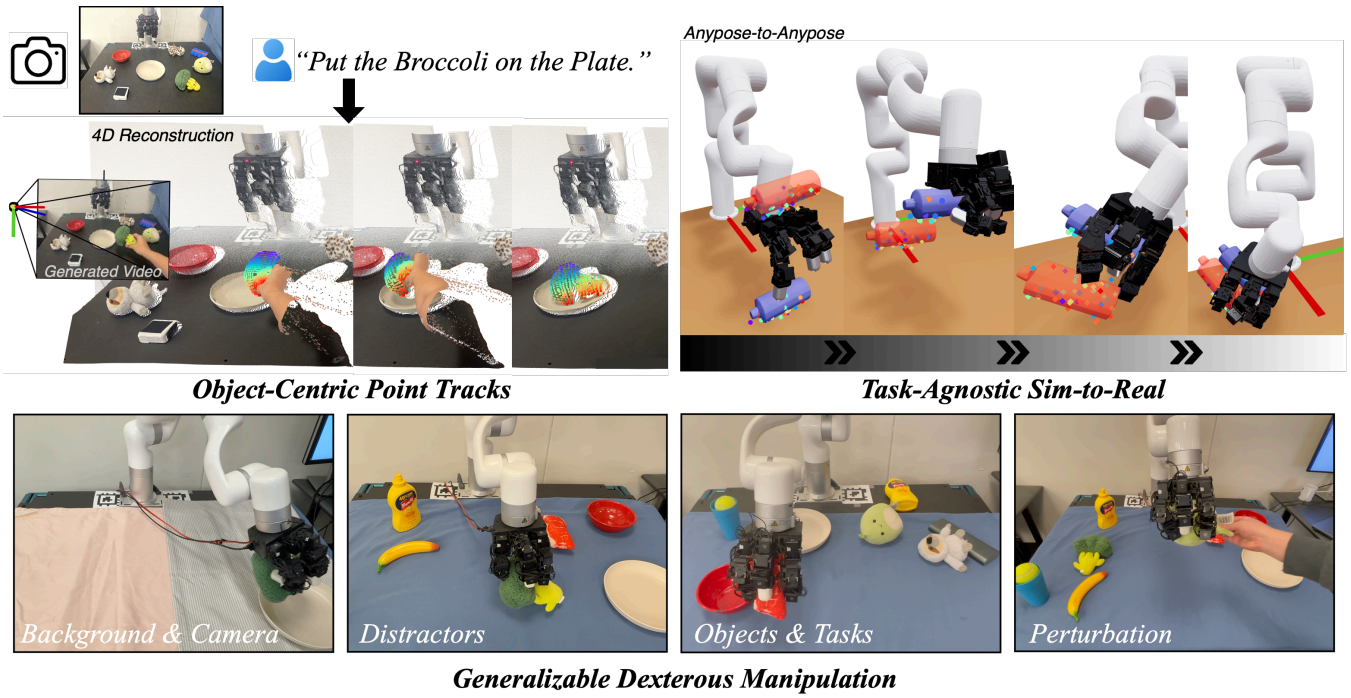

Fig. 1: Overview of Dex4D. We leverage video generation and 4D reconstruction to generate **object-centric point tracks**. Conditioned on the point tracks, we use a **task-agnostic sim-to-real policy** trained via Anypose-to-Anypose for task execution. Our policy trained entirely in simulation can be seamlessly deployed in the real world and generalizes to diverse configurations.

I. Introduction

The lack of high-quality, diverse, and scalable data remains a fundamental bottleneck in learning dexterous robot manipulation. Collecting real-world manipulation trajectories is expensive, difficult to instrument, and limited in coverage and diversity. Furthermore, learning dexterous manipulation via teleoperation poses unique challenges due to the difficulty of precisely controlling high-dimensional robotic hands and fingers, which makes large-scale data collection slow and error-prone [1].

Learning dexterous manipulation behaviors via sim-to-real reinforcement learning (RL) provides a promising alternative [2], [3]. Benefiting from highly parallel GPU-based simulation [4], [5] that largely improves interaction data bandwidth, RL-based policiess can be learnt in a few hours in simulation that are equivalent to years in the real world. However, training language-instructable 'generalist' robot policies in simulation requires substantial engineering effort, including designing complex simulation environments, specifying task descriptions and instructions, performing tedious reward shaping, and tuning RL pipelines across an ever-growing set of tasks [6], [7].

We argue that instead of learning language-conditioned and task-specific policies, we can use highly parallel simulation to learn fundamental **task-agnostic** manipulation skills that can be flexibly composed using a high-level planner, such as video generation models [8], [9] that have shown remarkable open-world generalization, to perform general downstream tasks. We operationalize this insight in our framework Dex4D, which learns a **point track conditioned** policy for Anypose-to-Anypose – *manipulating any object from any current pose to any target pose*. A key technical contribution lies in our goal representation – instead of separately encoding current and target object points, we propose **Paired Point Encoding** that leverages the correspondences across them.

We train our Anypose-to-Anypose policy across **thousands of objects** in simulation. The training process spans a broad space of object poses, trajectories, and hand-object interactions, enabling compositional generalization at test time. We show that our Anypose-to-Anypose policy, with geometry-aware and domain-robust point track representa-

tion as conditions, can be combined with video generation models to allow sim-to-real dexterous manipulation for generic tasks. Specifically, given a task description, Dex4D queries a foundational video model to generate a successful video task plan. We then leverage 4D reconstruction to extract object-centric point tracks from the generated video as an interface for goal specifications and policy conditions for our task-agnostic Anypose-to-Anypose policy, while using efficient online point tracking for closed-loop perception and control. As a result, Dex4D enables zero-shot transfer to real-world tasks **without any real robot finetuning**.

We evaluate Dex4D extensively in simulation and on real robotic platforms, comparing against state-of-the-art baselines. Our results (see Sec. A) show substantial improvements in success rate, task progress, and robustness. Furthermore, we demonstrate **strong generalization** to novel objects and poses, scene layouts, backgrounds, and task trajectories, highlighting the scalability and robustness of the proposed representation and learning framework. In summary, our contributions are as follows:

- We propose Anypose-to-Anypose, a task-agnostic sim-to-real learning formulation without tedious simulation tuning and task-specific reward shaping.
- We propose to leverage point tracks from generated videos and 4D reconstruction as an interface for goal specifications and policy conditions.
- We propose Paired Point Encoding, an effective goal representation, along with a point track conditioned transformer-based action world model architecture to improve policy learning.
- Extensive experiments demonstrate superior performance and strong generalization to unseen objects and poses, scene layouts, backgrounds, and task trajectories.

II. Learning Point Track Policy via Task-Agnostic Sim-to-Real

In this section, we introduce our point track policy trained via **Anypose-to-Anypose (AP2AP)**, a task-agnostic sim-to-real learning formulation for dexterous manipulation. We detail our AP2AP setup (§ II-A), our proposed Paired Point Encoding (§ II-B and Fig. 3), and teacher-student policy learning (§ II-C and Fig. 2). We then outline how to deploy the sim-to-real AP2AP policy using point tracks from generated videos in § III.

A. Anypose-to-Anypose

Anypose-to-Anypose (AP2AP) is a task-agnostic sim-to-real learning formulation for dexterous manipulation. AP2AP abstracts manipulation as directly transforming an object from an arbitrary initial pose to an arbitrary target pose in 3D space, without assuming task-specific structure, predefined grasps, or motion primitives. Unlike prior approaches [10], [11] that decompose manipulation into grasp generation, pose estimation, and planning, AP2AP treats object pose transformation itself as the fundamental learning objective, enabling a unified and reactive control policy for high-DoF dexterous hands.

We formulate AP2AP as a goal-conditioned Markov Decision Process (MDP) $\mathscr{M} = \langle \mathscr{S}, \mathscr{A}, \mathscr{T}, \mathscr{R}, \gamma, \mathscr{G} \rangle$ of state $s \in \mathscr{S}$, action $a \in \mathscr{A}$, transition function $\mathscr{T}$, reward $r \in \mathscr{R}$, discount factor γ, and goal $g \in \mathscr{G}$. The objective is to maximize the expected return $\mathbb{E}[\sum_t \gamma^t r_t]$ by finding an optimal policy $\pi^*(a_t|s_t, g_t)$, where the subscript t indexes the timestep.

At the beginning of each episode, an object is placed on the table with a random initial position and orientation. The first goal requires the robot to grasp and lift the object to a specified pose. Once a goal is stably achieved, the next goal is randomly set as a nearby target pose, encouraging continuous pose-to-pose transitions and effective local exploration.

Our AP2AP policy is trained entirely in simulation using **3,200 objects** from UniDexGrasp [2] under diverse pose configurations and extensive domain randomization. By learning to perform arbitrary pose-to-pose transformations on a wide range of objects, the policy acquires embodiment grounding and contact-rich manipulation skills in a task-agnostic manner. As a result, our method doesn't require task-specific tuning and generalizes zero-shot to unseen objects and downstream manipulation tasks in the real world.

B. Goal Representation via Paired Point Encoding

A key design choice for training the AP2AP policy is selecting a goal representation that can be robustly extracted at deployment time while remaining informative for pose-conditioned control. In this work, we represent objects using sparse object points, which are widely used, geometry-aware, and can be reliably obtained in the real world using point trackers. We can also easily obtain target object points given the desired transformation. Then a key design question is how to encode current and target object points as an effective goal representation so that they are maximally useful for policy learning.

A common approach is to encode current and target object points into two latent features and condition the policy on these features [12]. However, such encodings discard **correspondence** between current and target object points, which is critical for differentiating object poses. For example, when a ball undergoes pure rotation without translation, the shape of the points remains unchanged, even though the object pose is different. In this case, correspondence is the only information that distinguishes between identical shapes under different poses. To address this limitation, we propose **Paired Point Encoding**, a representation that explicitly preserves correspondence between the current and target object points. As illustrated in Fig. 3, given the current object points $\{\boldsymbol{p}_t^i\}_{i=1}^N$ and the target object points $\{\bar{\boldsymbol{p}}_t^i\}_{i=1}^N$ at timestep t, we construct paired points $\{\boldsymbol{q}_t^i\}_{i=1}^N$ by concatenating each pair of corresponding points. Each pair point is therefore 6-dimensional, consisting of a 3D current object point and its 3D target counterpart. For point index i at timestep t, the paired point is defined as below:

$$\boldsymbol{q}_t^i = \begin{bmatrix} \boldsymbol{p}_t^i \\ \bar{\boldsymbol{p}}_t^i \end{bmatrix} \in \mathbb{R}^6, \tag{1}$$

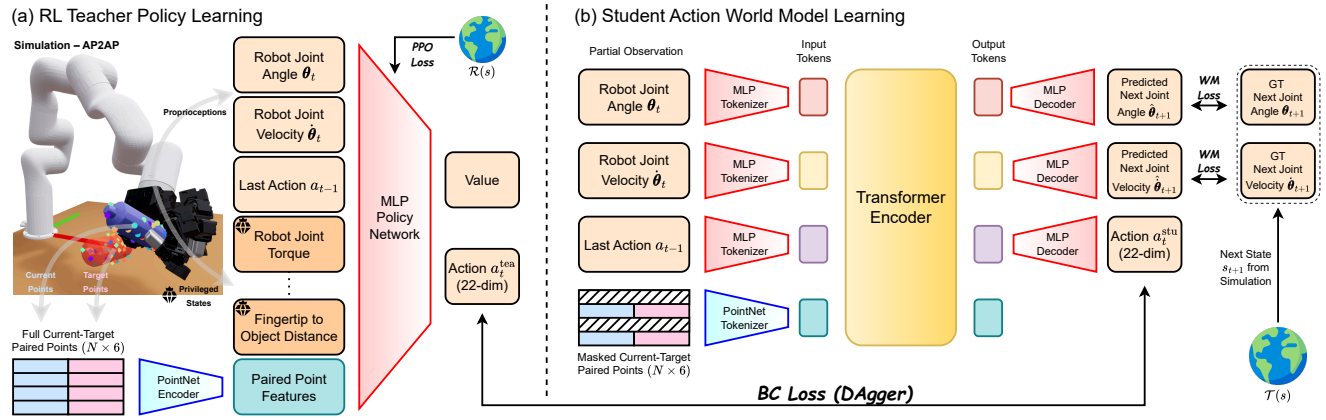

Fig. 2: Overview of our Dex4D teacher and student network architectures. **(a)** We first learn a teacher policy via RL with privileged states and full points sampled on the whole object, leveraging our proposed **Paired Point Encoding** representation. **(b)** Given partial observation, *i.e.,* robot proprioception, last action, and masked paired points, we distill from the teacher and learn a **transformer-based student action world model** that jointly predicts actions and future robot states.

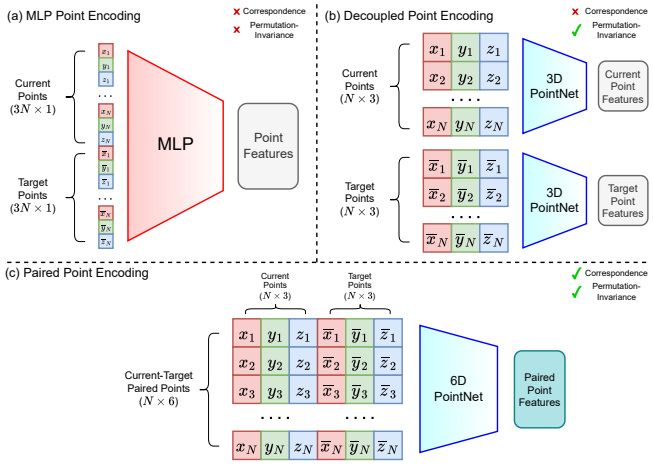

Fig. 3: Comparison between our Paired Point Encoding with other representations. Point features encoded from our Paired Point Encoding keep **correspondence** and **permutation-invariance** of the current and target object points, which shows better performance for policy learning.

These paired points $\{\boldsymbol{q}_t^i\}_{i=1}^N \in \mathbb{R}^{N\times 6}$ are then fed into a PointNet-style encoder [13], [14], which consists of shared MLP layers and mean-max mixed pooling to encode them into paired point features. In this way, we keep both **correspondence** and **permutation-invariance** of these points. Building on the Paired Point Encoding as goal representation, we now describe how it is used to train the AP2AP policy in simulation via a teacher–student learning framework.

C. Teacher-Student Policy Learning

To train the AP2AP policy in simulation, we follow a standard teacher-student distillation framework [15], [16]. As shown in Fig. 2, we first learn a teacher policy via visual RL [17] with proprioception, last action, privileged states and points uniformly sampled on the whole object, leveraging the Paired Point Encoding. Then, given only proprioception, last action, and partial points by masking, we leverage DAgger [18] to distill teacher to the student policy.

*1) **RL Teacher Policy Learning**:* In the first phase, we train a teacher policy using PPO [17] with privileged states and fully observed object geometry in simulation. As shown in Fig. 2, the state s_t consists of robot proprioception (joint angles and velocities), the last action, and privileged information (e.g., joint torques, fingertip-to-object distances, etc.). Following Sec. II-B, we compose current and target object points into paired points as the goal representation g_t and encode them using a lightweight PointNet [13] to preserve both correspondence and permutation invariance. The resulting feature is concatenated with the state components and provided as input to the PPO actor and critic networks.

To facilitate effective exploration and stable RL training, we adopt a **three-stage curriculum**. In the first stage, training is restricted to a single object category with a low environment reset threshold and a high robot arm speed limit to encourage early reward acquisition. In the second stage, the arm speed limit is reduced for real-world safety, and the reset threshold is increased. In the third stage, we train on all 3,200 objects with more challenging initializations and resets, lower control frequency, and more conservative learning updates. Throughout training, we apply extensive domain randomization, including observation and action noise, PD gains, hand–object friction, and external force disturbances, to enable smooth and robust sim-to-real transfer.

For reward shaping, instead of directly using the 6D pose (object position + rotation), our reward function leverages object points for a smoother reward landscape [19]. These rewards encourage the current object points to closely match the target object points, while promoting hand-object affinity and discouraging exaggerated motions:

$$r = r_{\text{goal}} + r_{\text{f,o}} + r_{\text{h,o}} + r_{\text{bonus}} + r_{\text{curl}} + r_{\text{table}} + r_{\text{action}} \qquad (2)$$

where r_{goal}, $r_{\text{f,o}}$, $r_{\text{h,o}}$, r_{bonus}, r_{curl}, r_{table}, and r_{action} represent rewards for current-target **point distances**, finger-object distances, hand-object distance, success bonus, finger curl, table collision penalty, and action penalty, respectively. More

details on curriculum design and reward shaping are provided in the supplementary materials.

*2) **Student Action World Model Learning**:* After training the teacher policy, we distill it into a student policy under partial observability using DAgger [18]. We introduce a **transformer-based action world model** that jointly learns action prediction and robot joint dynamics. This joint formulation improves action learning [12], [20]–[22] and supports safer and more controllable deployment, particularly for high-DoF and highly dynamic hand–arm systems.

As illustrated in Fig. 2, the student policy takes as input robot proprioception (joint angles and velocities), the last action, and masked paired points. These inputs are first tokenized using MLPs and a PointNet-style encoder [13], [14] with mean–max mixed pooling, and then processed by self-attention layers [23]. The output token corresponding to the last action a_{t-1} is used to decode the robot action a_t, while the tokens corresponding to the current joint angle $\boldsymbol{\theta}_t$ and velocity $\dot{\boldsymbol{\theta}}_t$ are used to predict the next-state joint angle $\boldsymbol{\theta}_{t+1}$ and velocity $\dot{\boldsymbol{\theta}}_{t+1}$.

The action world model is trained with a combination of a DAgger behavior cloning loss and a world modeling loss:

$$\begin{aligned}\mathscr{L} &= \mathscr{L}_{\text{bc}} + \mathscr{L}_{\text{wm}} \\ &= \|a_t^{\text{stu}} - a_t^{\text{tea}}\|_1 + \left\| \begin{bmatrix} \hat{\boldsymbol{\theta}}_{t+1} - \boldsymbol{\theta}_{t+1} \\ \hat{\dot{\boldsymbol{\theta}}}_{t+1} - \dot{\boldsymbol{\theta}}_{t+1} \end{bmatrix} \right\|_1 \end{aligned} \tag{3}$$

where a_t^{stu} and a_t^{tea} denote the student and teacher actions, respectively, and $\hat{\boldsymbol{\theta}}_{t+1}$, $\hat{\dot{\boldsymbol{\theta}}}_{t+1}$ and $\boldsymbol{\theta}_{t+1}$, $\dot{\boldsymbol{\theta}}_{t+1}$ denote the predicted and ground-truth next-step joint angles and velocities.

To emulate object point occlusions caused by the hand in real-world settings and improve robustness to monocular viewpoints, we introduce a **random plane-height masking strategy**. Specifically, we sample a random plane through the object center and mask out points on one side of the plane, followed by sampling a random height that masks the majority of points above it and a minority below it. Target points are masked accordingly based on correspondence. This strategy enables the student policy to generalize across diverse camera viewpoints under partial observation. Further implementation details are provided in the supplementary materials.

III. Generalizable Dexterous Manipulation from Point Track Policy

To leverage the AP2AP policy in the real world, we condition it on desired 3D point tracks – a sequence of target object points that specifies the desired object configuration over time. Such point tracks can be obtained from diverse sources, including video generation models or one-shot human demonstrations. In this section, we describe how we extract point tracks from generated videos and how they are used to drive closed-loop policy execution.

A. From Generated Video to Object-Centric Point Tracks

Large-scale video generation models provide a rich source of object motion and manipulation information, as they are trained on vast collections of Internet videos [10], [11], [24], [25]. To use these videos for real-world robot deployment, we lift the videos into object-centric point tracks, which define a target trajectory that can directly condition the AP2AP policy.

Formally, given a language instruction l and an initial RGBD observation $\{I_0, D_0\}$, we first generate a sequence of future RGB frames $\{I_t\}_{t=1}^{T}$ using an off-the-shelf video generation model. Using the initial object segmentation mask and the generated frames, we first perform **2D point tracking** to obtain object 2D point tracks $\{\bar{\boldsymbol{u}}_t^i\}_{t=1,i=1}^{T,N} \in \mathbb{R}^{T\times N\times 2}$.

Next, we perform **relative depth estimation** for each frame and calibrate it using the initial depth observation D_0. Specifically, each estimated depth map is scaled based on the ratio between the median depth of the frame and the median depth of the initial observation. This allows us to lift the 2D point tracks into metric 3D point tracks $\{\bar{\boldsymbol{p}}_t^i\}_{t=1,i=1}^{T,N} \in \mathbb{R}^{T\times N\times 3}$. The object-centric point tracks serve as goal specifications of the task and target object points to condition the policy.

The resulting point tracks provide a structured, object-centric representation of the desired pose over time. This representation can be directly used as a plan for the AP2AP point track policy, guiding the robot to perform pose-to-pose manipulation in the real world. Compared to prior approaches that rely on fully metric depth estimation and explicit calibration [10], [11], calibrating relative depth produces smoother and more stable metric depths, resulting in cleaner point tracks. Additional implementation details and visualizations are provided in the supplementary materials.

B. Closed-Loop Perception and Control

With the target point tracks defined, we now describe how the AP2AP policy executes it in a closed loop in the real world.

At the start of execution, the first set of target points is assigned to the policy. The initial tracked 2D points from the generated video are also provided for an online point tracker [26], which is used to track the object 2D points from the image observation in real time, which are then back-projected to 3D using the RGBD camera.

At each timestep, the current object 3D points and target object 3D points are composed into paired points and provided to the student policy along with robot proprioception and the last action. The policy then outputs actions for robot control. To determine when to advance to the next set of target points, we compute the average distance between corresponding visible points at each timestep:

$$d_t = \frac{1}{N'} \sum_{i=1}^{N'} \|\boldsymbol{p}_t^i - \bar{\boldsymbol{p}}_t^i\|_2 \tag{4}$$

where N' is the number of visible points. When d_t falls below a threshold, the target points are updated to the next ones in the goal 3D point tracks. This process repeats in a closed loop until the final target points $\{\bar{\boldsymbol{p}}_T^i\}_{i=1}^{N}$ are reached.

REFERENCES

[1] Zhao-Heng Yin, Changhao Wang, Luis Pineda, Francois Hogan, Krishna Bodduluri, Akash Sharma, Patrick Lancaster, Ishita Prasad, Mrinal Kalakrishnan, Jitendra Malik, et al. Dexteritygen: Foundation controller for unprecedented dexterity. *arXiv preprint arXiv:2502.04307*, 2025.

[2] Yinzhen Xu, Weikang Wan, Jialiang Zhang, Haoran Liu, Zikang Shan, Hao Shen, Ruicheng Wang, Haoran Geng, Yijia Weng, Jiayi Chen, et al. Unidexgrasp: Universal robotic dexterous grasping via learning diverse proposal generation and goal-conditioned policy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4737–4746, 2023.

[3] Yuanpei Chen, Yiran Geng, Fangwei Zhong, Jiaming Ji, Jiechuang Jiang, Zongqing Lu, Hao Dong, and Yaodong Yang. Bi-dexhands: Towards human-level bimanual dexterous manipulation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(5):2804–2818, 2023.

[4] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

[5] Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, Antonio Serrano-Muñoz, Xinjie Yao, René Zurbrügg, Nikita Rudin, et al. Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning. *arXiv preprint arXiv:2511.04831*, 2025.

[6] Tairan He, Zi Wang, Haoru Xue, Qingwei Ben, Zhengyi Luo, Wenli Xiao, Ye Yuan, Xingye Da, Fernando Castañeda, Shankar Sastry, et al. Viral: Visual sim-to-real at scale for humanoid loco-manipulation. *arXiv preprint arXiv:2511.15200*, 2025.

[7] Yufei Wang, Zhou Xian, Feng Chen, Tsun-Hsuan Wang, Yian Wang, Katerina Fragkiadaki, Zackory Erickson, David Held, and Chuang Gan. Robogen: Towards unleashing infinite data for automated robot learning via generative simulation. *arXiv preprint arXiv:2311.01455*, 2023.

[8] Thaddäus Wiedemer, Yuxuan Li, Paul Vicol, Shixiang Shane Gu, Nick Matarese, Kevin Swersky, Been Kim, Priyank Jaini, and Robert Geirhos. Video models are zero-shot learners and reasoners. *arXiv preprint arXiv:2509.20328*, 2025.

[9] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

[10] Shivansh Patel, Shraddhaa Mohan, Hanlin Mai, Unnat Jain, Svetlana Lazebnik, and Yunzhu Li. Robotic manipulation by imitating generated videos without physical demonstrations. *arXiv preprint arXiv:2507.00990*, 2025.

[11] Hongyu Li, Lingfeng Sun, Yafei Hu, Duy Ta, Jennifer Barry, George Konidaris, and Jiahui Fu. Novaflow: Zero-shot manipulation via actionable flow from generated videos. *arXiv preprint arXiv:2510.08568*, 2025.

[12] Jiangran Lyu, Ziming Li, Xuesong Shi, Chaoyi Xu, Yizhou Wang, and He Wang. Dywa: Dynamics-adaptive world action model for generalizable non-prehensile manipulation. *arXiv preprint arXiv:2503.16806*, 2025.

[13] 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.

[14] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[15] Tyler Ga Wei Lum, Martin Matak, Viktor Makoviychuk, Ankur Handa, Arthur Allshire, Tucker Hermans, Nathan D Ratliff, and Karl Van Wyk. Dextrah-g: Pixels-to-action dexterous arm-hand grasping with geometric fabrics. *arXiv preprint arXiv:2407.02274*, 2024.

[16] Ritvik Singh, Arthur Allshire, Ankur Handa, Nathan Ratliff, and Karl Van Wyk. Dextrah-rgb: Visuomotor policies to grasp anything with dexterous hands. *arXiv preprint arXiv:2412.01791*, 2024.

[17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[18] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[19] Siheng Zhao, Yanjie Ze, Yue Wang, C Karen Liu, Pieter Abbeel, Guanya Shi, and Rocky Duan. Resmimic: From general motion tracking to humanoid whole-body loco-manipulation via residual learning. *arXiv preprint arXiv:2510.05070*, 2025.

[20] Yuxuan Kuang, Qin Han, Danshi Li, Qiyu Dai, Lian Ding, Dong Sun, Hanlin Zhao, and He Wang. Stopnet: Multiview-based 6-dof suction detection for transparent objects on production lines. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5389–5396. IEEE, 2024.

[21] Chuning Zhu, Raymond Yu, Siyuan Feng, Benjamin Burchfiel, Paarth Shah, and Abhishek Gupta. Unified world models: Coupling video and action diffusion for pretraining on large robotic datasets. *arXiv preprint arXiv:2504.02792*, 2025.

[22] Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, et al. Worldvla: Towards autoregressive action world model. *arXiv preprint arXiv:2506.21539*, 2025.

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[24] Zhiting Mei, Tenny Yin, Ola Shorinwa, Apurva Badithela, Zhonghe Zheng, Joseph Bruno, Madison Bland, Lihan Zha, Asher Hancock, Jaime Fernández Fisac, et al. Video generation models in robotics-applications, research challenges, future directions. *arXiv preprint arXiv:2601.07823*, 2026.

[25] Boyuan Chen, Tianyuan Zhang, Haoran Geng, Kiwhan Song, Caiyi Zhang, Peihao Li, William T Freeman, Jitendra Malik, Pieter Abbeel, Russ Tedrake, et al. Large video planner enables generalizable robot control. *arXiv preprint arXiv:2512.15840*, 2025.

[26] Nikita Karaev, Yuri Makarov, Jianyuan Wang, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker3: Simpler and better point tracking by pseudo-labelling real videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6013–6022, 2025.

[27] Kenneth Shaw, Ananye Agarwal, and Deepak Pathak. Leap hand: Low-cost, efficient, and anthropomorphic hand for robot learning. *arXiv preprint arXiv:2309.06440*, 2023.

[28] Edwin Olson. Apriltag: A robust and flexible visual fiducial system. In *2011 IEEE international conference on robotics and automation*, pages 3400–3407. IEEE, 2011.

[29] Matei Ciocarlie, Corey Goldfeder, and Peter Allen. Dexterous grasping via eigengrasps: A low-dimensional approach to a high-complexity problem. In *Robotics: Science and systems manipulation workshop-sensing and adapting to the real world*, 2007.

[30] Ananye Agarwal, Shagun Uppal, Kenneth Shaw, and Deepak Pathak. Dexterous functional grasping. *arXiv preprint arXiv:2312.02975*, 2023.

[31] Sili Chen, Hengkai Guo, Shengnan Zhu, Feihu Zhang, Zilong Huang, Jiashi Feng, and Bingyi Kang. Video depth anything: Consistent depth estimation for super-long videos. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 22831–22840, 2025.

[32] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024.

[33] Berk Calli, Aaron Walsman, Arjun Singh, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. Benchmarking in manipulation research: The ycb object and model set and benchmarking protocols. *arXiv preprint arXiv:1502.03143*, 2015.

[34] Adithyavairavan Murali, Balakumar Sundaralingam, Yu-Wei Chao, Wentao Yuan, Jun Yamada, Mark Carlson, Fabio Ramos, Stan Birchfield, Dieter Fox, and Clemens Eppner. Graspgen: A diffusion-based framework for 6-dof grasping with on-generator training. *arXiv preprint arXiv:2507.13097*, 2025.

[35] Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Foundations of Crystallography*, 32(5):922–923, 1976.

[36] Yuxuan Kuang, Haoran Geng, Amine Elhafsi, Tan-Dzung Do, Pieter Abbeel, Jitendra Malik, Marco Pavone, and Yue Wang. Skillblender:

Towards versatile humanoid whole-body loco-manipulation via skill blending. *arXiv preprint arXiv:2506.09366*, 2025.

[37] Toru Lin, Kartik Sachdev, Linxi Fan, Jitendra Malik, and Yuke Zhu. Sim-to-real reinforcement learning for vision-based dexterous manipulation on humanoids. *arXiv preprint arXiv:2502.20396*, 2025.

[38] Ritvik Singh, Karl Van Wyk, Pieter Abbeel, Jitendra Malik, Nathan Ratliff, and Ankur Handa. End-to-end rl improves dexterous grasping policies. *arXiv preprint arXiv:2509.16434*, 2025.

[39] Min Liu, Zherong Pan, Kai Xu, Kanishka Ganguly, and Dinesh Manocha. Deep differentiable grasp planner for high-dof grippers. *arXiv preprint arXiv:2002.01530*, 2020.

[40] Tengyu Liu, Zeyu Liu, Ziyuan Jiao, Yixin Zhu, and Song-Chun Zhu. Synthesizing diverse and physically stable grasps with arbitrary hand structures using differentiable force closure estimator. *IEEE Robotics and Automation Letters*, 7(1):470–477, 2021.

[41] Ruicheng Wang, Jialiang Zhang, Jiayi Chen, Yinzhen Xu, Puhao Li, Tengyu Liu, and He Wang. Dexgraspnet: A large-scale robotic dexterous grasp dataset for general objects based on simulation. *arXiv preprint arXiv:2210.02697*, 2022.

[42] Jialiang Zhang, Haoran Liu, Danshi Li, XinQiang Yu, Haoran Geng, Yufei Ding, Jiayi Chen, and He Wang. Dexgraspnet 2.0: Learning generative dexterous grasping in large-scale synthetic cluttered scenes. In *8th Annual Conference on Robot Learning*, 2024.

[43] Zhao-Heng Yin and Pieter Abbeel. Lightning grasp: High performance procedural grasp synthesis with contact fields. *arXiv preprint arXiv:2511.07418*, 2025.

[44] Chen Wang, Haochen Shi, Weizhuo Wang, Ruohan Zhang, Li Fei-Fei, and C Karen Liu. Dexcap: Scalable and portable mocap data collection system for dexterous manipulation. *arXiv preprint arXiv:2403.07788*, 2024.

[45] Toru Lin, Yu Zhang, Qiyang Li, Haozhi Qi, Brent Yi, Sergey Levine, and Jitendra Malik. Learning visuotactile skills with two multifingered hands. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5637–5643. IEEE, 2025.

[46] Ruoshi Wen, Guangzeng Chen, Zhongren Cui, Min Du, Yang Gou, Zhigang Han, Liqun Huang, Mingyu Lei, Yunfei Li, Zhuohang Li, et al. Gr-dexter technical report. *arXiv preprint arXiv:2512.24210*, 2025.

[47] Weikang Wan, Haoran Geng, Yun Liu, Zikang Shan, Yaodong Yang, Li Yi, and He Wang. Unidexgrasp++: Improving dexterous grasping policy learning via geometry-aware curriculum and iterative generalist-specialist learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3891–3902, 2023.

[48] Haozhi Qi, Ashish Kumar, Roberto Calandra, Yi Ma, and Jitendra Malik. In-hand object rotation via rapid motor adaptation. In *Conference on Robot Learning*, pages 1722–1732. PMLR, 2023.

[49] Haozhi Qi, Brent Yi, Sudharshan Suresh, Mike Lambeta, Yi Ma, Roberto Calandra, and Jitendra Malik. General in-hand object rotation with vision and touch. In *Conference on Robot Learning*, pages 2549–2564. PMLR, 2023.

[50] Jun Wang, Ying Yuan, Haichuan Che, Haozhi Qi, Yi Ma, Jitendra Malik, and Xiaolong Wang. Lessons from learning to spin" pens". *arXiv preprint arXiv:2407.18902*, 2024.

[51] Xueyi Liu, He Wang, and Li Yi. Dexndm: Closing the reality gap for dexterous in-hand rotation via joint-wise neural dynamics model. *arXiv preprint arXiv:2510.08556*, 2025.

[52] Zhengyi Luo, Jinkun Cao, Sammy Christen, Alexander Winkler, Kris Kitani, and Weipeng Xu. Omnigrasp: Grasping diverse objects with simulated humanoids. *Advances in Neural Information Processing Systems*, 37:2161–2184, 2024.

[53] Xueyi Liu, Jianibieke Adalibieke, Qianwei Han, Yuzhe Qin, and Li Yi. Dextrack: Towards generalizable neural tracking control for dexterous manipulation from human references. *arXiv preprint arXiv:2502.09614*, 2025.

[54] Sirui Xu, Yu-Wei Chao, Liuyu Bian, Arsalan Mousavian, Yu-Xiong Wang, Liangyan Gui, and Wei Yang. Dexplore: Scalable neural control for dexterous manipulation from reference scoped exploration. In *Conference on Robot Learning*, pages 2184–2199. PMLR, 2025.

[55] Austin Patel, Andrew Wang, Ilija Radosavovic, and Jitendra Malik. Learning to imitate object interactions from internet videos. *arXiv preprint arXiv:2211.13225*, 2022.

[56] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, et al. Video generation models as world simulators. *OpenAI Blog*, 1(8):1, 2024.

[57] Homanga Bharadhwaj, Abhinav Gupta, Vikash Kumar, and Shubham Tulsiani. Towards generalizable zero-shot manipulation via translating human interaction plans. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6904–6911. IEEE, 2024.

[58] Homanga Bharadhwaj, Roozbeh Mottaghi, Abhinav Gupta, and Shubham Tulsiani. Track2act: Predicting point tracks from internet videos enables generalizable robot manipulation. In *European Conference on Computer Vision*, pages 306–324. Springer, 2024.

[59] Yuxuan Kuang, Junjie Ye, Haoran Geng, Jiageng Mao, Congyue Deng, Leonidas Guibas, He Wang, and Yue Wang. Ram: Retrieval-based affordance transfer for generalizable zero-shot robotic manipulation. *arXiv preprint arXiv:2407.04689*, 2024.

[60] Mengda Xu, Zhenjia Xu, Yinghao Xu, Cheng Chi, Gordon Wetzstein, Manuela Veloso, and Shuran Song. Flow as the cross-domain manipulation interface. *arXiv preprint arXiv:2407.15208*, 2024.

[61] Homanga Bharadhwaj, Debidatta Dwibedi, Abhinav Gupta, Shubham Tulsiani, Carl Doersch, Ted Xiao, Dhruv Shah, Fei Xia, Dorsa Sadigh, and Sean Kirmani. Gen2act: Human video generation in novel scenarios enables generalizable robot manipulation. *arXiv preprint arXiv:2409.16283*, 2024.

[62] Sungjae Park, Homanga Bharadhwaj, and Shubham Tulsiani. Demodiffusion: One-shot human imitation using pre-trained diffusion policy. *arXiv preprint arXiv:2506.20668*, 2025.

[63] Hongyan Zhi, Peihao Chen, Siyuan Zhou, Yubo Dong, Quanxi Wu, Lei Han, and Mingkui Tan. 3dflowaction: Learning cross-embodiment manipulation from 3d flow world model. *arXiv preprint arXiv:2506.06199*, 2025.

[64] Karthik Dharmarajan, Wenlong Huang, Jiajun Wu, Li Fei-Fei, and Ruohan Zhang. Dream2flow: Bridging video generation and open-world manipulation with 3d object flow. *arXiv preprint arXiv:2512.24766*, 2025.

[65] Haoyi Zhu, Yating Wang, Di Huang, Weicai Ye, Wanli Ouyang, and Tong He. Point cloud matters: Rethinking the impact of different observation spaces on robot learning. *Advances in Neural Information Processing Systems*, 37:77799–77830, 2024.

[66] Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. *arXiv preprint arXiv:2403.03954*, 2024.

[67] Tsung-Wei Ke, Nikolaos Gkanatsios, and Katerina Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. *Arxiv*, 2024.

[68] Nikolaos Gkanatsios, Jiahe Xu, Matthew Bronars, Arsalan Mousavian, Tsung-Wei Ke, and Katerina Fragkiadaki. 3d flowmatch actor: Unified 3d policy for single- and dual-arm manipulation. *Arxiv*, 2025.

[69] Yanjie Ze, Ge Yan, Yueh-Hua Wu, Annabella Macaluso, Yuying Ge, Jianglong Ye, Nicklas Hansen, Li Erran Li, and Xiaolong Wang. Gnfactor: Multi-task real robot learning with generalizable neural feature fields. In *Conference on robot learning*, pages 284–301. PMLR, 2023.

[70] Weiheng Liu, Yuxuan Wan, Jilong Wang, Yuxuan Kuang, Xuesong Shi, Haoran Li, Dongbin Zhao, Zhizheng Zhang, and He Wang. Fetchbot: Learning generalizable object fetching in cluttered scenes via zero-shot sim2real. In *9th Annual Conference on Robot Learning*, 2025.

[71] Guanxing Lu, Shiyi Zhang, Ziwei Wang, Changliu Liu, Jiwen Lu, and Yansong Tang. Manigaussian: Dynamic gaussian splatting for multi-task robotic manipulation. In *European Conference on Computer Vision*, pages 349–366. Springer, 2024.

[72] Brent Yi, Chung Min Kim, Justin Kerr, Gina Wu, Rebecca Feng, Anthony Zhang, Jonas Kulhanek, Hongsuk Choi, Yi Ma, Matthew Tancik, et al. Viser: Imperative, web-based 3d visualization in python. *arXiv preprint arXiv:2507.22885*, 2025.

APPENDIX

A. Experiments

1) Implementation Details: In our work, we use a 22-DoF dexterous hand-arm system that comprises a 6-DoF xArm6 robot arm and a 16-DoF LEAP hand [27]. We use a single-view RealSense D435 camera for RGBD sensing and Apriltags [28] for calibration.

For the action space, the policy outputs a 22-dimensional action, which contains 6-DoF arm delta joint angles and 16-DoF LEAP hand absolute joint angles. We found that this configuration is more robust in RL training without compromising on full hand dexterity [15], [29], [30]. The action is then converted to the 22-DoF joint target used for motor control.

For desired point tracks acquisition, we use Wan2.6 [9] for video generation, Video Depth Anything [31] for relative video depth estimation, SAM2 [32] for initial frame segmentation, and CoTracker3 [26] for 2D point tracking. All these components are modular, reusable, and replaceable, which can be updated and swapped to new state-of-the-art models.

We train teacher and student policies using the Isaac Gym simulator [4]. For point numbers, we sample 128 points for teacher training, and 64 points for student training and inference. For RL teacher training, we use symmetric PPO [17] to learn 15k steps for the first stage, 10k steps for the second stage, and another 25k steps for the third stage to ensure convergence. For student policy learning, we select the best teacher checkpoint and learn 25k steps using DAgger [18]. It takes 2-3 days for teacher training, and around 20 hours for student training, using a single NVIDIA RTX A6000 GPU.

2) Simulation Experiments: We evaluate our method with several baselines in a simulation suite to provide a fair and reproducible comparison.

*a) **Tasks**:* We evaluate our method and baselines on six dexterous manipulation tasks, namely *Apple2Plate*, *Pour*, *Hammer*, *StackCup*, *RotateBox*, and *Sponge2Bowl*. These tasks involve dexterous grasping, arm movement, object reorientation, and spatial reasoning. We select objects from our object set from UniDexGrasp [2] and the **unseen** YCB dataset [33]. For more details on task specifications, please refer to the supplementary materials.

*b) **Baselines**:* We compare against NovaFlow [11], a method that leverages 3D actionable point tracks for robot manipulation. NovaFlow first extracts 3D action point tracks from generated videos, and then applies a grasp generator [34] and performs open-loop motion planning based on transformation estimation between current and target object points using the Kabsch algorithm [35]. Since NovaFlow is parallel gripper-only, for fair comparison, we adapt it to dexterous hands by applying our method for dexterous grasping and locking the fingers after lifting. Then we conduct pose estimation based on the current state and perform motion planning in an open-loop manner.

Moreover, since NovaFlow [11] is open-loop without feedback, which is crucial for high-dynamics tasks like dexterous manipulation, we also implemented a closed-loop version of NovaFlow (NovaFlow-CL) as a baseline. Specifically, we first apply our method for dexterous grasping, and then at each timestep, we estimate the transformation between current and target object points, and do real-time IK to move the robot arm accordingly.

*c) **Evaluation Protocols and Metrics**:* In the simulation, we manually pre-define several waypoints as goal poses and compute the target points for policy condition. We also apply random Gaussian noise to both current and target object points independently for realism. The goal pose will update once the current goal pose is achieved.

For observation, instead of using masked points as input, we put an RGBD camera in the scene and compute real-time object points based on visibility.

To measure performance, we use Success Rate **(SR)** and Task Progress **(TP)** as quantitative metrics. Success is determined by whether the last waypoint is achieved or not, and Task Progress is defined as the average number of waypoints achieved. For each trial, we initialize the object with a random pose, and we conduct 100 trials for each task.

*d) **Results and Analysis**:* We report our simulation results in Table. I. As shown in the results, our method outperforms all baselines by a large margin, demonstrating the effectiveness and superiority of our method.

First by comparing open-loop NovaFlow [11] and closed-loop NovaFlow-CL, we can see that closed-loop feedback greatly improves performance (+9.2% in SR and 16% in TP) since the object may move around in the hand. Closed-loop pose estimation could perform replanning so that the object could remain on the correct track instead of accumulating pose errors, achieving a huge bonus on task progress.

Further comparing our method with NovaFlow-CL, we also surpass it, achieving a significant performance gain (+16.3% in SR and +10.4% in TP). This demonstrates that our policy, trained on AP2AP, exhibits strong generalizability to unseen tasks, objects, and trajectories, despite both methods utilizing closed-loop feedback. We attribute this performance gain to the fact that during the RL and DAgger process, the policy is able to traverse sufficient states so that it can generalize to unseen scenarios. NovaFlow-CL fails mainly because of insufficient hand reactivity to high dynamics that causes object falling, limited motion planning solutions that severely occlude object points, and pose estimation errors due to point noises, especially when the number of visible points is low. This further highlights our method's robustness and adaptivity to out-of-distribution scenarios and effectiveness in generalizing across objects and scenes.

3) Ablation Studies: To further analyze our framework design, we conduct ablation studies on various components to highlight the importance of each element in our method. Ablation results are shown in Table II and Fig. 4, showing that our method surpasses all ablations, proving the effectiveness of all the modules.

*a) **Importance of Paired Point Encoding**:* First, we discuss whether using our proposed Paired Point Encod-

Method	CL	*Apple2Plate*		*Pour*		*Hammer*		*StackCup*		*RotateBox*		*Sponge2Bowl*		Avg.	
		SR	TP	SR	TP	SR	TP	SR	TP	SR	TP	SR	TP	SR	TP
NovaFlow [11]	✗	0.44	0.57	0.66	0.76	0.12	0.16	0.42	0.59	0.25	0.32	0.18	0.29	0.345	0.448
NovaFlow-CL [11]	✓	0.58	0.76	0.73	0.85	0.27	0.30	0.44	0.73	0.33	0.60	0.27	0.41	0.437	0.608
`Dex4D` **(Ours)**	✓	**0.86**	**0.91**	**0.86**	**0.92**	**0.28**	**0.31**	**0.60**	**0.82**	**0.44**	**0.69**	**0.56**	**0.62**	**0.600**	**0.712**

TABLE I: Quantitative results measured by Success Rate **(SR)** and Task Progress **(TP)** in simulation. **CL** stands for closed-loop. We adapt NovaFlow and NovaFlow-CL to dexterous hands by using our method for first-stage dexterous grasping.

Method	SR	TP
MLP Point Encoding	0.057	0.172
Decoupled Point Encoding	0.203	0.363
w/o Self-Attention	0.490	0.675
w/o World Modeling	0.570	0.683
`Dex4D` **(Ours)**	**0.600**	**0.712**

TABLE II: Ablation studies on the student policy. **SR** and **TP** are averaged across tasks.

ing can improve the performance for both student and teacher policies. We compare against two variants as shown in Fig. 3 (a)(b): (a) *MLP Point Encoding* that directly tokenizes current and target object points using MLPs; (b) *Decoupled Point Encoding* that use two PointNet tokenizers to encode current and target object points separately, without using our proposed paired points. Results of student policy distillation and RL teacher training (for stage 1 & 2) are shown in Table II and Fig. 4, respectively. Results show that using our Paired Point Encoding significantly improve performance for both policies.

For the **student** policy, our finding suggests that using MLP to encode the points leads to severe performance degradation, making SR reduce to 5.7%. We also find out that using separate PointNet encoders to encode current and target object points individually will also severely lower the performance since the policy loses the correspondence between the two sets of points. This highlights our representation's advantage to effectively encode the object geometry information and point correspondence.

We also ablate with *MLP Point Encoding* and *Decoupled Point Encoding* in our RL **teacher** training to verify the effectiveness of our Paired Point Encoding representation in RL training. As shown in Fig. 4, our method outperforms both variants, demonstrating that our representation and framework can even boost the performance for visual RL, which is considered harder in prior works [36]–[38].

*b) **Ablation on Policy Architecture**:* Furthermore, we also ablate different neural network architecture design choices for our student policy. We compare against two other variants: (c) *w/o Self-Attention* that concatenates tokens and uses MLP to decode actions; and (d) *w/o World Modeling* that discards next state prediction. The results are shown in Table II, showing that our method outperforms both ablations, proving the effectiveness of our student policy design.

Compared with *w/o Self-Attention* that naïvely concatenates proprioception and paired point features and uses MLP to decode actions, self-attention layers (*i.e.,* transformer encoder) can attend to different tokens from proprioception and paired points, which better captures relations from different input components and achieves non-trivial performance gains. We also find that integrating world modeling can improve performance, which correlates well with the synergistic effects of policy learning and world modeling.

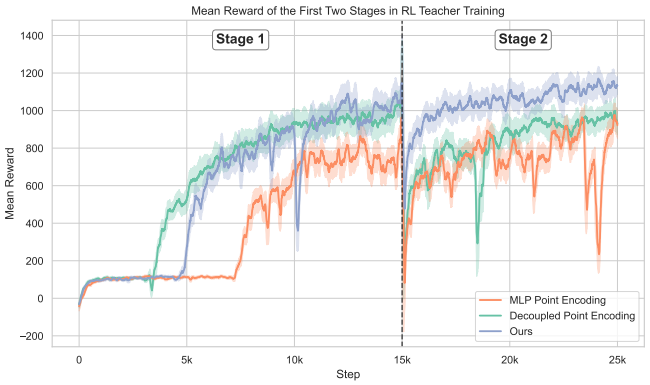

Fig. 4: Mean reward curve of the first two stages of teacher training. Step 15k is the curriculum boundary. Our method outperforms both ablation variants.

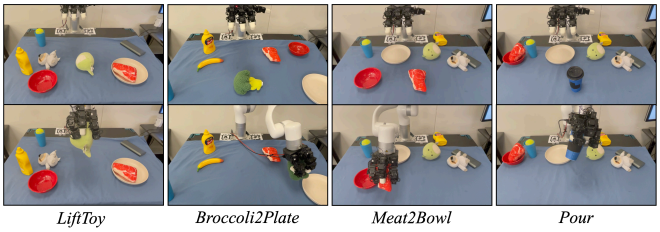

Fig. 5: Overview of real-world dexterous manipulation tasks. Two frames are shown in each column for each task.

4) Real-World Experiments: We deploy our simulation-trained policy to the real world and conduct extensive experiment suites on four tasks, namely *LiftToy*, *Broccoli2Plate*, *Meat2Bowl*, and *Pour*, as shown in Fig. 5. **Note that all the objects are unseen and there are no real robot demonstrations for any task.** We deploy our policy as the procedure detailed in Sec. III. We compare our method against NovaFlow-CL [11], which uses the real-time current and target object points to estimate 6D transformation at each planning step and do motion planning to reach the pose. Note that for NovaFlow-CL, we first leverage our method to grasp

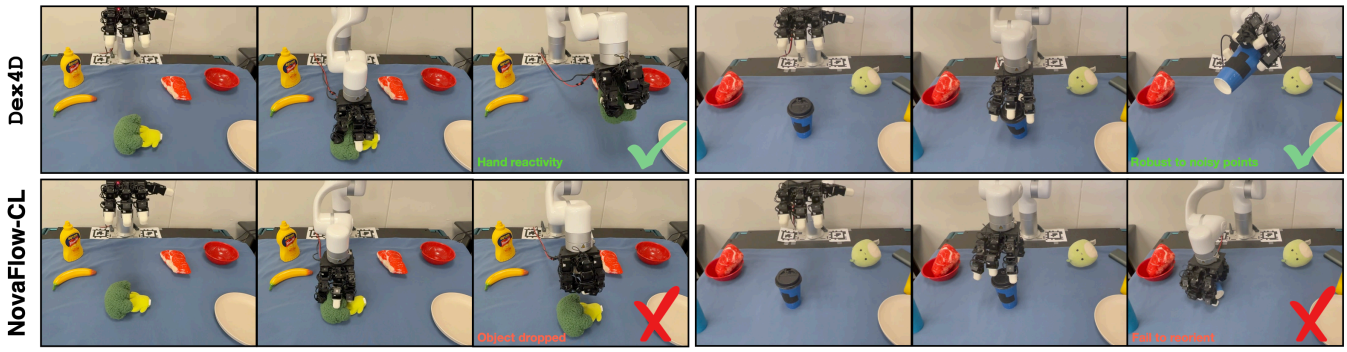

Fig. 6: Qualitative comparison between our method and the baseline. The baseline method suffers from object dropping and inaccurate post-grasping movement due to the **lack of hand feedback** and **vulnerability to few and noisy visible points**, while our method performs robustly.

Method	*LiftToy*	*Broccoli2Plate*	*Meat2Bowl*	*Pour*	Total
NovaFlow-CL [11]	4/10	3/10	3/10	0/10	10/40
`Dex4D` **(Ours)**	**6/10**	**4/10**	**5/10**	**4/10**	**19/40**

TABLE III: Quantitative results measured by Success Rate **(SR)** in the real world. All the objects are **unseen** and there are **no** real robot demonstrations.

the object and then perform closed-loop pose estimation, which is the same as in simulation experiments.

As shown in Table III, our method achieves a 22.5% performance gain in **SR** compared to the baseline, demonstrating our method's superiority against the motion planning-based method. The superiority of our method mainly comes from the closed-loop reactivity for both arm and hand, and robust action prediction under noisy point input. Video results and comparisons can be found in the supplementary video.

We also show the qualitative comparison in Fig. 6 and the supplementary video. As we can see, the baseline fails due to a couple of reasons. First, since the baseline is unaware of the hand and object grasping, the object would gradually fall off the hand during arm moving due to the lack of feedback. In contrast, our method learns to adjust or regrasp the object and then proceeds with the task. Moreover, in the real world the 3D point tracking poses large amounts of noise, including ones coming from inaccurate 2D tracking, noisy depth sensing, and latency, especially when the LEAP hand fingers severely occlude the object. Since the baseline method leverages the Kabsch algorithm [35] to solve the 6D pose, it's prone to noisy observation especially when visible points are few. The Kabsch algorithm can hardly solve the correct rotation between noisy current and target object points under real-world scenarios when the object is occluded by the hand, as in the *Pour* task, the baseline has a 0 success rate. In contrast, our method remains robust even if there are less than 10 visible points left. Finally, some failures of the baseline come from limited solutions of motion planning since we use a 6-DoF xArm6, which completely occludes the object from the camera.

Moreover, our method is robust to various generalization tests in the real world. As shown in Fig. 1, although our policy is only trained on single-object scenarios purely in simulation, it generalizes well to unseen object types and poses, backgrounds, camera views, task trajectories, and external disturbances.

However, we also noticed some failure modes of our policy, which can be further improved in future works. First, in the real world, the real-time CoTracker3 [26] will lose track of the object when there are significant object movements, similar nearby textures, or unintended object rotation that blocks initially tracked points. This is the major cause of failures. Sometimes the policy also tends to push the object to form a firm grasp, but this might pose extra forces that in turn knocks over the object.

B. Related Works

1) Generalizable Dexterous Manipulation: Endowing robots with human-level and generalizable dexterity is a long-standing goal for generalist robots that work under diverse real-world scenarios. It's also very challenging due to its high-DoF and high-dynamics nature. Prior optimization-based works often rely on contact-based optimization [39]–[43] to synthesize dexterous grasping poses, which are executed by motion planning. However, these works are mainly limited to grasping and prone to disturbances without closed-loop feedback. Another line of works uses mocap devices or teleoperation to collect dexterous manipulation data and train policies on them via imitation learning [44]–[46]. However, these works suffer from in-domain data collection and fail to generalize to unseen tasks, objects, and scenes. Recently, reinforcement learning (RL) has shown promise on generalizable dexterous manipulation, including dexterous grasping [2], [15], [16], [38], [47], in-hand reorientation [1], [48]–[51], and motion tracking [52]–[55]. Nonetheless, they often lack autonomy for high-level tasks that require task-specific planning. In contrast to all these works, our method leverages video generation and 4D reconstruction for high-level planning and trains an object-centric task-agnostic

policy that works across tasks using sim-to-real RL with point tracks as an interface, achieving generalizable and autonomous dexterous manipulation.

2) Video-Based Robot Learning: Recent years witnessed huge progress in video generation [8], [9], [56] and learning from human videos [57]–[62]. Video generation models not only can be used for entertainment or simulation, but also serve as world models or powerful high-level planners for robotics tasks since they are trained on enormous amounts of Internet videos and contain rich human priors [24]. Recent works [10], [11], [25], [63], [64] leverage video generation models or flow models as planners and use either pose estimation with motion planning or heuristic retargeting to map generated pixels to actions. However, these works suffer from large embodiment gaps and a lack of closed-loop feedback, which are crucial for highly dynamic tasks such as dexterous manipulation. They also require either object mesh [10] or clean point tracks [11] for pose estimation, which is hard to satisfy in the real world, especially with finger occlusions. In contrast, we train a closed-loop policy via sim-to-real, leveraging our proposed Paired Point Encoding representation along with extensive point masking and domain randomization. Therefore, our method is robust to real-world noisy sensor input and can generalize to diverse unseen configurations.

3) 3D Policy Learning: Spatial understanding is crucial for robot agents to reason about the 3D scene around us. Therefore, it's important to find a good 3D representation for policy learning. [65]–[68] leverage point cloud as input for imitation learning, and [66] proves the sufficiency of minimal PointNet [13] to encode the point cloud. Others use scene representations (voxelized neural fields [69], occupancy [70], and Gaussian Splatting [71]) for policy learning. Compared to these works, our work extends goal-conditioned policy learning by using 3D representations as goal conditions. We propose Paired Point Encoding as policy conditions that combine current object points with target object points, supporting task-agnostic learning without specific language instructions as conditions. We also leverage world modeling as auxiliary supervision signals to jointly learn action prediction and robot dynamics from proprioception and 3D perception.

C. Conclusions, Limitations, and Future Works

Conclusions. In this work, we propose `Dex4D`, a framework for generalizable dexterous manipulation via object-centric point tracks and task-agnostic sim-to-real learning. At the core of `Dex4D` is to decouple recognition and control by leveraging video generation and 4D reconstruction to generate object-centric point tracks as high-level planning, and training a task-agnostic sim-to-real policy for low-level control. We further propose a novel Paired Point Encoding representation and a transformer-based action world model to enhance 3D goal-conditioned policy learning. Extensive experiments in simulation and the real world verify the effectiveness of our framework, and show our remarkable generalization to unseen tasks, objects, and scenes. We hope our work can benefit future research on generalizable dexterous manipulation.

Limitations and Future Works. Despite compelling results, our work has certain limitations that can be further improved in future works. First, in our work we didn't incorporate human grasp priors from HOI datasets and Internet videos due to the lack of amount and diversity of clean mocap sequences and the large embodiment gap between human hands and the LEAP hand, which is large in size and only has four thick fingers. It's promising to leverage these abundant hand-related data sources along with thinner and more human-like dexterous hands to unlock more functional behaviors. Second, our AP2AP formulation is currently limited to single-object manipulation. Extending it to objects with more complicated geometries, such as articulated objects, would be a promising direction. Additionally, how to incorporate other modalities, such as tactile sensing, is also an interesting question. Finally, in the future we could develop more accurate, robust, and faster online tracking models for better point tracking that enables lower latency and better tracking performance on the deployment side.

D. Videos and Visualizations

Videos and visualization results can be found in our supplementary video. We use Viser [72] for all the visualizations. We thank the authors for their great work.

E. Task Specifications

In simulation, we evaluate our method and baselines on six dexterous manipulation tasks, namely *Apple2Plate*, *Pour*, *Hammer*, *StackCup*, *RotateBox*, and *Sponge2Bowl*. We illustrate these tasks in Fig. 7. Their task objectives are:

- *Apple2Plate*: Grasp an apple on the table and put it on the plate.
- *Pour*: Grasp the mug on the table and tilt to pour.
- *Hammer*: Grasp the hammer on the table and strike forward.
- *StackCup*: Pick up the cup and stack it onto another cup on the table.
- *RotateBox*: Pick up the Foam Box on the table and rotate it horizontally 90 degrees in the air.
- *Sponge2Bowl*: Grasp a thin piece of sponge on the table and put it into the bowl.

In the real world, we evaluate on four tasks, namely *LiftToy*, *Broccoli2Plate*, *Meat2Bowl*, and *Pour*, as shown in Fig. 5. Their task objectives are:

- *LiftToy*: Grasp a toy on the table and lift it to a certain pose in the air.
- *Broccoli2Plate*: Pick up the broccoli on the table and put it on the plate.
- *Meat2Bowl*: Pick up the meat on the table and put it into the bowl.
- *Pour*: Grasp the coffee cup on the table and tilt to pour.

F. Video Generation and Point Track Extraction

For video generation, we use Wan2.6 [9] with its online platform, and use its native prompt enhancement with Chinese prompts, which show better performance than English

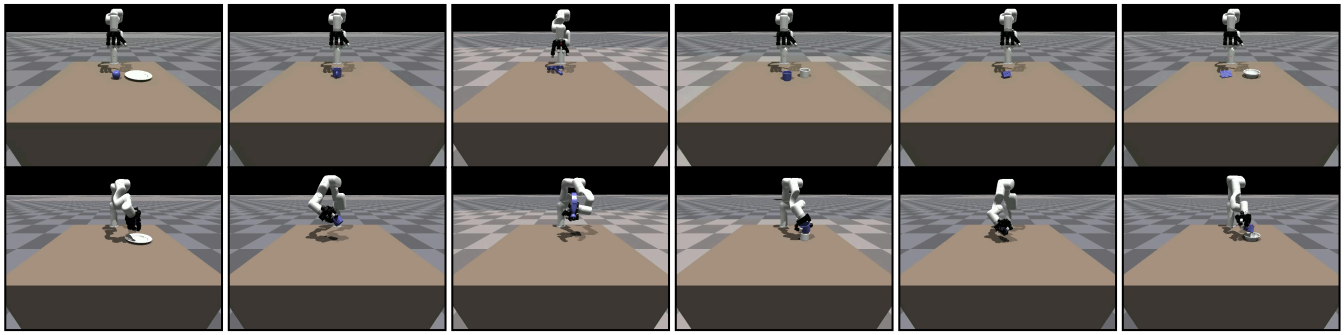

Fig. 7: Overview of simulated tasks. Two frames are shown in each column for each task.

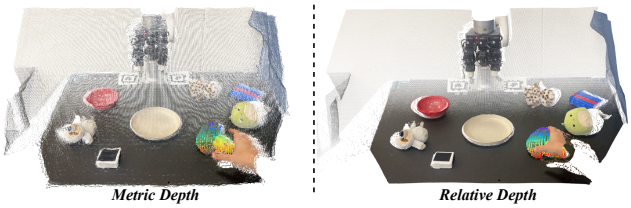

Fig. 8: Comparison between metric depth estimation and relative depth estimation. Relative depth estimation yields smoother, more spatio-temporally consistent results and fewer floater points.

prompts [11]. We use Wan's first frame + language prompt conditioned generation mode and generate 5-second 30-FPS 720P videos.

For 3D point track extraction, we find that using relative depth estimation, rather than metric video depth estimation as in prior work [11], [55], yields better results, with greater spatio-temporal consistency and fewer floaters, as shown in Fig. 8.

G. RL State Space

The state space $\mathscr{S}$ for the RL teacher policy training includes: joint angles, joint velocities, the last action, joint torques, fingertip states (*state* denotes 6D pose, linear and angular velocities, hereinafter the same), fingertip forces, hand state, object state, goal pose, 64-dimensional object point cloud feature encoded by a pretrained PointNet [2], [13], and fingertip-to-object distance vectors.

H. RL Curriculum Learning

We detail our three-stage curriculum in Table IV.

I. Reward Function

We detail our reward shaping in Table V. p_j^{finger}, p^{obj}, p^{hand}, h_j^{finger} represent the 3D position of the finger j, the 3D position of the object, the 3D position of the hand palm and the height of the finger j, respectively. And conditions contact, success, and stay_success are defined as follows:

	Stage 1	Stage 2	Stage 3
Step	0-15k	15k-25k	25k-50k
Object Category	Bottle	Bottle	All
Too Far Reset Threshold (m)	0.3	1×10^6	1×10^6
Goal Reset Stable Ratio	0.1	0.2	0.2
Arm Speed Scale	10	1.5	1.5
Control Frequency (Hz)	30	30	5
Step Size	3×10^{-4}	3×10^{-4}	3×10^{-5}
Initial Hand Position	Low	Low	High

TABLE IV: Curriculum settings of our three-stage RL teacher training.

Term	Expression	Weight (Stage 1 & 2)	Weight (Stage 3)
r_{goal}	$\mathbb{1}_{\text{contact}}(1.4 - 3d_t)$	1.0	1.0
$r_{\text{f,o}}$	$-0.5 \times \sum_{j=1}^{4} d(p_j^{\text{finger}}, p^{\text{obj}})$	1.0	0.0
$r_{\text{h,o}}$	$-0.5 \times d(p^{\text{hand}}, p^{\text{obj}})$	1.0	0.0
r_{bonus}	$\mathbb{1}_{\text{contact}} \mathbb{1}_{\text{success}} \frac{5.0}{1+10d_t} + \mathbb{1}_{\text{stay\_success}} \times 10$	1.0	1.0
r_{curl}	$-0.001 \lVert \boldsymbol{\theta}_t \rVert_2^2$	1.0	10.0
r_{table}	$\min(10 \min_j(h_j^{\text{finger}} - 0.62), 0)$	1.0	1.0
r_{action}	$-0.01 \lVert a_t \rVert_2^2$	1.0	5.0

TABLE V: Reward shaping for RL teacher training in different stages.

$$\text{contact} = \begin{cases} 1, & \sum_{j=1}^{4} d(p_j^{\text{finger}}, p^{\text{obj}}) < 0.48\,\text{m} \\ & \text{and } d(p^{\text{hand}}, p^{\text{obj}}) < 0.12\,\text{m} \\ 0, & \text{otherwise} \end{cases}$$

$$\text{success} = \begin{cases} 1, & d_t < 0.05\,\text{m} \\ 0, & \text{otherwise} \end{cases}$$

$$\text{stay\_success} = \begin{cases} 1, & \text{success\_time} \geq 0.5\,\text{s} \\ 0, & \text{otherwise} \end{cases}$$

J. Domain Randomization and External Perturbation

Isaac Gym [4] offers a suite of domain randomization functions for RL training. We detail our domain randomization setup in Table VI.

Term	Operation	Distribution	Range
Observation White Noise	Additive	Gaussian	$[0, 0.002]$
Observation Correlated Noise	Additive	Gaussian	$[0, 0.001]$
Action White Noise	Additive	Gaussian	$[0, 0.05]$
Action Correlated Noise	Additive	Gaussian	$[0, 0.015]$
Gravity	Additive	Gaussian	$[0, 0.4]$
Joint Stiffness	Scaling	Uniform	$[0.9, 1.1]$
Joint Damping	Scaling	Uniform	$[0.9, 1.1]$
Joint Lower Limit	Additive	Gaussian	$[0, 0.01]$
Joint Upper Limit	Additive	Gaussian	$[0, 0.01]$
Robot Mass	Scaling	Uniform	$[0.5, 1.5]$
Robot Friction	Scaling	Uniform	$[0.7, 1.3]$
Object Mass	Scaling	Uniform	$[0.5, 1.5]$
Object Friction	Scaling	Uniform	$[0.7, 1.3]$

TABLE VI: Domain Randomization Setup.

In addition to physics parameter randomization, we also perform a random object force pushing mechanism to improve the policy's robustness to external perturbation. Specifically, we implement the pushing as linear and angular velocities applied to the object. Every four seconds in the simulation, we apply a linear xy velocity $\sim \mathscr{U}(-0.2, 0.2)$ and an angular velocity $\sim \mathscr{U}(-0.2, 0.2)$ to the object.

K. Paired Point Masking

As in Sec. II-C, we introduce a **random plane-height masking strategy** for the student policy learning to improve our policy's robustness to real-world point input.

In student learning, for each environment, we first perform **plane masking**. Specifically, we randomly sample one plane that crosses the object's centroid, select one side of it, and then mask out all the points on that side. In this way, we obtain approximately half of the original object points. These point indices are kept the same throughout the whole environment. This process is to simulate the single-view observation in our real-world deployment so that our policy can generalize to varying camera views.

For the remaining points, at each timestep, we then apply **height masking**. Specifically, we first randomly sample a height ratio in $[0.2, 1.0]$, and based on the height, we mask out 90% of the points above this height and 5% of the points below. Finally, we apply a Gaussian noise $\sim \mathscr{N}(0, 0.005)$ to the remaining points. This process is to simulate the occlusion between fingers and object points so that our policy can generalize to fewer object points and point noises in the real world.

L. Additional Hyperparameters

Additional hyperparameters of the teacher and student policies are detailed in Table VII and Table VIII, respectively.

Hyperparameter	Value
Algorithm	PPO
Optimizer	Adam
Number of Environments	4096
Number of Object Points	128
Discount Factor (λ)	0.95
GAE parameter (γ)	0.96
Desired KL Divergence	0.016
Clip Range	0.2
Actor MLP Hidden Dimension	$[1024, 1024, 512, 512]$
Critic MLP Hidden Dimension	$[1024, 1024, 512, 512]$
PointNet Hidden Dimension	$[128, 128]$

TABLE VII: Hyperparameters of the Teacher Policy.

Hyperparameter	Value
Algorithm	DAgger
Optimizer	Adam
Number of Environments	3200
Number of Object Points	64
Learning Rate	1×10^{-4}
Token Dimension	128
Number of Self-Attention Layers	4

TABLE VIII: Hyperparameters of the Student Policy.