Learning Particle-Based World Model from Human for Robot Dexterous Manipulation

Zhengdong Hong*, Yulin Liu*, Haowen Hou, Bo Ai, Jun Wang, Tongzhou Mu, Yuzhe Qin, Jiayuan Gu, Hao Su

***Abstract*—Human data constitute a rich repository of knowledge about environment dynamics and embodied interactions. Existing methods attempt to use this data by retargeting human motion to robots or pretraining visual representations. However, these approaches may not generalize to novel object configurations or capture the underlying dynamics of interactions. Our key insight is that the dynamics of human-object interactions are transferable to robots and can guide reinforcement learning (RL) by reducing the exploration space. Specifically, we learn a particle-based simplified dynamics model from human data to generate diverse high-level trajectories through model predictive control (MPC). Then those trajectories can successfully guide reinforcement learning (RL) in physical simulations by improving sample efficiency and success rate, enabling generalizable manipulations after Sim-to-Real transfer. Extensive real-world experiments demonstrate our method not only outperforms baselines in higher success rate and better generalizability for single-object grasping tasks but also succeeds in challenging cluttered environments where existing methods fail, establishing an effective framework for human-guided dexterous manipulation.** [1]

I. Introduction

Human-level dexterous manipulation in robots is essential for advancing robotic capabilities. However, dexterous manipulation is challenging due to the high degrees of freedom of robot hands and the complex non-linear dynamics in hand-object interactions. Previous approaches either rely on model-based trajectory optimization or model-free reinforcement learning (RL). Model-based trajectory optimization [22, 15] struggles to handle the complexity of dynamics. Reinforcement learning methods [25, 11, 2, 7, 31] highly rely on task-specific reward designs and are inefficient with high-dimensional exploration spaces. Although imitation learning methods [10] trained on real robot data showcase a solution to the problem, they require time-consuming collection of large amounts of real robot data[30, 3] which is not feasible for laboratories. Compared to that, human data is rich in quantity and it implicitly encodes human experience in manipulation, which shows great potential in learning dexterous manipulation.

What should we learn from the human data? Since there is a large morphology gap between humans and robots, directly applying human data to robots is not feasible. Some previous work [19, 18, 21] seek to learn visual features from human videos, while the pre-trained representation often relies on real-robot data for fine-tuning. Others choose to learn from human motion trajectories. Existing methods [23, 8, 9, 27] use optimization methods (e.g. inverse kinematics) for retargeting human trajectories to robot motions, and perform imitation learning or RL to follow the trajectories. However, they are essentially memorizing and replaying human trajectories, failing to generalize or adapt to unseen scenarios.

Our core idea is to build a simplified physical world to better leverage the abundant human experience and priors in human data for guiding robot learning. In this world, complex physics (force, mass, contact) are abstracted away, while the successful human-object interaction priors (motion, collision) are preserved. Specifically, we learn a particle-based dynamics model from the human-object interaction datasets. Having this generalizable world model, we are able to search for diverse high-level trajectories according to different task and object configurations. Remarkably, since our model is learned from human data which by default contains all-success demonstrations, it is easy to obtain a plausible trajectory by searching in our world, which contains the human experience and dynamics priors necessary for learning complex dexterous manipulation skills efficiently. Our key insights are as follows:

- Our model facilitates efficient robot planning. By removing complex low-level physics which is not necessary for high-level planning, our model could generate reasonable planning trajectories in seconds, reducing the extensive overhead of directly exploring in the real physical world or in physical simulations[34, 28] for learning dexterous manipulation.
- By constructing a simplified physical world by human experience, we can adaptively search for diverse trajectories according to different environment and object configurations in it. Compared to previous methods, we have better generalizability to novel environment and object configurations.
- Our two-stage pipeline is efficient. The high-level trajectories generated by the world model can facilitate low-level skill reinforcement learning in physical simulations. We successfully demonstrate this in simulation and real robots.

II. Methods

A. Overview

Our goal is to enable dexterous hand grasping that can quickly generalize to complex scenarios, including unseen objects and cluttered environments. To achieve this, we propose a two-stage pipeline that first learns to generate high-quality reference trajectories through particle-based dynamics modeling and model predictive control, and then trains policies in simulation using reinforcement learning (RL) guided by the generated trajectories. The first stage serves as high-level planning and enables quick adaptiveness to novel environments with physics abstraction, whereas the second stage recovers physics detail and enables sim-to-real transfer.

[1] *denote equal contribution

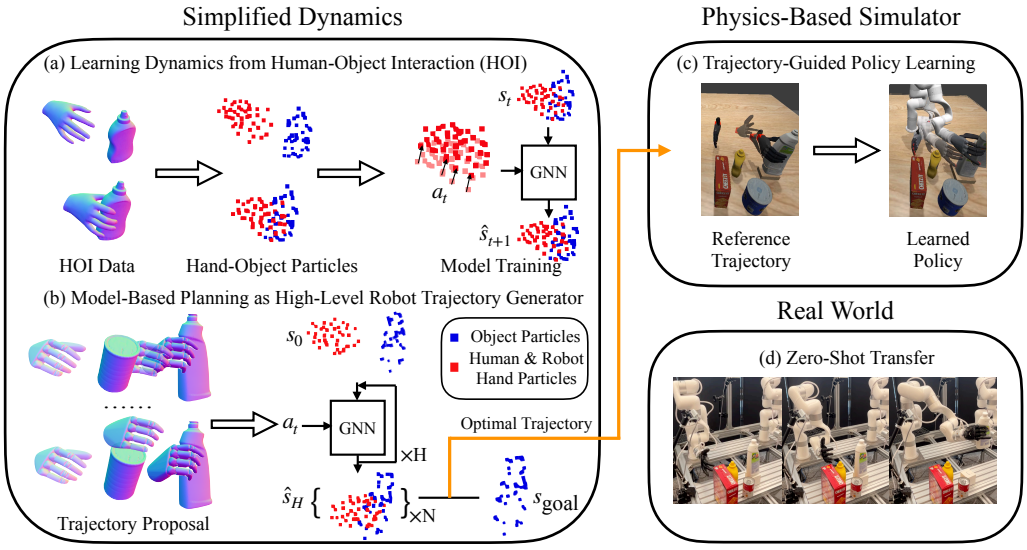


Fig. 1. **Method Overview**.We employ a two-stage pipeline that enables dexterous hand grasping in complex, unseen environments. (a) A particle-based dynamics model is learned from human-object interaction data. (b) The shared particle-based representation between human and robotic hands enables model-based planning for robot trajectory generation (c) The generated trajectories are used to guide reinforcement learning in a physics-based simulator to recover detailed dynamics. (d) The learned policy is then transferred zero-shot to the real world.

We formalize this problem as a Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$. At each time step t, the robot observes the state $s_t \in \mathcal{S}$ and selects an action a_t according to a policy π: $a_t \sim \pi(a_t|s_t)$. The environment then transits to state s_{t+1} and the robot receives a reward r_t. The policy is optimized to maximize reward given a reference trajectory τ_i.

The reference trajectory τ_i is generated by minimizing a cost function between desired goal state with a sequence of states predicted by dynamics model f. The model trained on human-object interaction data captures hand-object dynamics priors and operates in a particle-based representation of hand and object, which is shared between human and robotic hands.

B. Learning World Model from Human Object Interactions

We argue that particle representation is a transferable representation for real-world hand-object interactions. Specifically, it enables adapting dynamics priors learned from human hands to robot hands or novel objects by modeling local particle-based behaviors that generalize across different entities.

We train a GNN [17, 1] on DexYCB dataset [5], which contains real-world 3D hand and object pose sequences. Hand poses are represented using MANO model [26], while objects are with scanned meshes and 6DoF poses. To construct particle-based representations, we sample a fixed set of vertices from the MANO hand template and uniformly sample surface points from each object mesh, maintaining temporal correspondence by applying the poses at each frame.

Our model operates in the joint hand and object particle position space $s_t = (x_t^{\text{obj}}, x_t^{\text{hand}})$, taking as input the current state s_t and action a_t defined as displacement of hand particle positions $a_t = x_{t+1}^{\text{hand}} - x_t^{\text{hand}}$, and predicts the next state s_{t+1}.

At each timestep t, we construct a graph $G_t = \langle V_t, E_t \rangle$ where each node $v_{i,t} = \langle x_{i,t}, c_{i,t}, a_{i,t} \rangle$ encodes the particle's position $x_{i,t}$, particle attribution $c_{i,t} \in 0, 1$ (where $c = 0$ denotes object particles and $c = 1$ denotes hand particles), and action $a_{i,t}$. For hand particles, the action is defined as the displacement between consecutive frames and is set to zero for object particles. Intra-hand and intra-object edges are pre-defined based on spatial proximity in a canonical pose, while hand-object edges are dynamically constructed by connecting particles within a distance threshold. Please refer to the appendix C for additional details.

Given the graph, we first obtain node feature and edge feature through node decoder and edge encoder respectively. These features are thus propagated through edges using message passing, and the updated node features are decoded to predict the next-step particle positions $x_{i,t+1}$.

To reduce error accumulation over time, we train the model autoregressively: starting from the ground-truth initial state s_0, we recursively predict future states, using each predicted state as input for the next prediction. The training loss is the mean squared error (MSE) between predicted and ground-truth particle positions over a prediction horizon H:

$$\mathcal{L} = \frac{1}{H} \sum_{t=1}^{H-1} \|\hat{s}_t - s_t\|_2^2 \tag{1}$$

C. MPC for Searching High-level Trajectories

With the learned dynamics model, we apply Model Predictive Control (MPC) to optimize robot hand trajectories by minimizing a cost function over predicted states. Specifically, we leverage Predictive Path Integral (MPPI)[33] to iteratively sample and refine action sequences.

To address the complexity of planning over long horizons, we sample a sequence of key-frame robot commands $\{u_t\}_{t=0}^{H'}$ and interpolate them to generate a full action sequence over the horizon H. These commands are converted into hand particle trajectories via forward kinematics, ensuring compatibility with the particle-based dynamics model. To maintain temporal

consistency, we pre-sample particles on the surface of the robot hand in a canonical pose and update their positions at each timestep via forward kinematics.

Planning proceeds as follows: given an initial object configuration, we sample a batch of robot commands $\{u_t\}_{t=0}^{H'}$, which are converted into hand particle trajectories. These trajectories, together with the current object state, are used as inputs to the learned dynamics model to predict future object states autoregressively. Each predicted trajectory is evaluated by a cost function based on the predicted object and hand configurations. We then re-weight the action samples using importance sampling and update the distribution parameters accordingly. This process is repeated iteratively, and the action sequence with the lowest expected cost is selected for execution.

We use a weighted cost function defined as

$$\omega_1 \mathcal{L}_{\text{pc}} + \omega_2 \mathcal{L}_{\text{finger}} + \omega_3 \mathcal{L}_{\text{palm}} + \omega_4 \mathcal{L}_{\text{pene}} + \omega_5 \mathcal{L}_{\text{attr}} \quad (2)$$

Specifically, $\mathcal{L}_{\text{pc}}$ measures mean squared error (MSE) between the predicted and goal object particles, encouraging the object to reach target configuration. $\mathcal{L}_{\text{finger}}$ and $\mathcal{L}_{\text{palm}}$ penalize distance between hand and the object to promote reaching behavior. $\mathcal{L}_{\text{pene}}$ discourages hand-object penetration, while $\mathcal{L}_{\text{attr}}$ encourages stable grasp postures. Please refer to Appendix D-A for detailed loss definitions.

D. Reinforcement Learning for Low-level Policies

Due to limited data distribution, the learned dynamics model cannot fully capture complex physical interactions but simplified dynamics behaviours, resulting in inaccurate robot-hand-object interaction – "magic" grasping. Though imperfect, these high-level trajectories contain meaningful information like approaching behaviors which can guide RL in a physical simulator to obtain physically plausible trajectories.

We train a state-based policy which takes robot joint positions and 6D object poses as input states and predicts delta end-effector poses and delta hand joint positions. During control, the end-effector pose is converted to arm joint positions using inverse kinematics (IK) using Pinocchio [4].

Reward function consists of 4 items weighted by $\lambda_{1,2,3,4}$:

$$R = \lambda_1 R_f + \lambda_2 R_{\text{contact}} + \lambda_3 R_{\text{lifting}} + \lambda_4 R_{\text{success}} \quad (3)$$

The first term, following reward R_f, defined as $R_f = \beta_1 R_j + \beta_2 R_{\text{ee},x} + \beta_3 R_{\text{ee},r}$, encourages the robot to follow reference hand motions. The other terms are designed to encourage task success. Please check the appendixA-C for detailed illustrations of each item.

III. Experiments

In this section, we investigate the following questions:

- Does our generative reference trajectory enable sample-efficient learning of grasping policies on both *seen* and *unseen* objects?
- Can our framework generalize to cluttered environments and perform obstacle avoidance without explicit collision supervision?
- Can the policy learned in simulation successfully transfer to the real world?

A. Sample-Efficient Grasping on Seen and Unseen Objects

1) Experiment Setup: **Task Setup.** For single-object grasping task, we divide objects into 3 test sets: 1. DexYCB [5] objects used in dynamics model training 2. DexYCB [5] objects **not** used in dynamics model training 3. **In-the-wild** daily objects. For each object, we sample a reference trajectory and train a separate policy. Performance is evaluated using the success rate (SR), defined as whether the object reaches a predefined height. Each experiment is repeated three times with different random seeds, and we report the average performance metrics. Please see simulation environment and policy implementation details in appendix A-A.

Baselines. We compare our method with 3 competitive baselines: (i)**RL Engineering** replacing the trajectory-following term replaces by reaching reward to encourage the hand reaching the object defined as where $R_r = 1 - \tanh\left(\alpha_1 \left\| T^t_{\text{palm}} - T^t_{\text{object}} \right\|_2\right)$ where T^t_{palm} and T^t_{object} are the 3D position of the palm link on the robot hand and the 3D position of the object at timestep t, respectively. (ii) **Human Motion Retargeting** using human-demonstrated trajectories from the DexYCB dataset with motion retargeting [24] from human hand to robotic hand, (iii) **S2S Trajectory Generator (State-to-State Regression)** uses a GPT2-style transformer to predict the future hand-object state S_{t+10} using the history hand-object states $\mathbf{S_{t+a}}(\mathbf{a} = 0, 1, ..., 9)$ in an auto-regressive manner. Note that the only difference between the baselines and our method lies **only** in the choice of reference trajectory; all other settings for policy training are kept consistent.

2) Results: Table I summarizes results for single-object grasping tasks. It should be noted that ***seen*** and ***unseen*** refer exclusively to whether an object was included in the training data for the **dynamics model** only. Our method demonstrates strong consistency across both categories, indicating that the learned dynamics model generalizes well to unseen objects with minimal performance degradation.

While the Human-Retargeting baseline benefits from high-quality human demonstration data and achieves a strong overall success rate, ours performs comparably, without relying on additional data for novel objects. Moreover, we outperform the other two baselines by a large margin.

As shown in Figure 3, our method converges faster even compared with Human-Retargeting baseline. This efficiency advantage is likely due to the embodiment gap between human and robotic hands in motion retargeting [24], which can result in penetrations or infeasible grasp poses.

B. Obstacle Avoidance in Cluttered Environments

1) Evaluation Setup: **Task Setup.** We create three clustered environments with either a large obstacle or densely packed objects to evaluate generalization to unseen scenarios, making obstacle avoidance during execution highly challenging. In each environment, we define two different target objects for grasping, resulting in a total of six distinct scenarios. Performance is evaluated using the success rate (SR), defined as the percentage of successful lifts where the obstacle is untouched.

SR (%) ↑ Ref. Trajectory	*Seen YCB Objects* `mustard`	`cracker`	*Unseen YCB Objects* `sugar`	`bleach`	*Unseen In-the-Wild Objects* `cerave`	`chocolate`	`gelsheet`	`moisture`	Avg.
None	33	6	22	16	3	0	8	12	13
Human-Motion-Retargeting	78	61	-	-	-	-	-	-	-
Trajectory Generator	64	58	42	27	0	28	42	0	33
Ours	**82**	**69**	**81**	**90**	**69**	**53**	**83**	**58**	**73**

TABLE I

QUANTITATIVE RESULTS IN SIMULATION: *seen* AND *unseen* REFER TO WHETHER AN OBJECT IS IN THE TRAINING DATA OF THE DYNAMICS MODEL.

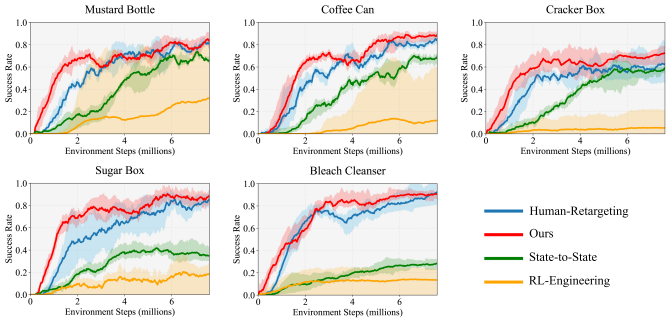


Fig. 3. Learning curves (shaded areas show variances of the 3 seeds).

Each experiment is repeated three times with different random seeds, and we report the average performance metrics.

Implementation. We reuse the dynamics model trained in the previous experiment section but modify the trajectory generation process by incorporating obstacle information. Specifically, we extend the hand-object penetration cost in MPC to account for penetration with each individual object. During the second stage of RL training (Section II-D), our policy follows the generated obstacle-avoidance trajectory as a reference, without direct access to explicit obstacle information. For a fair comparison, we introduce **an extra penalty** reward item **only** for all baseline methods to directly access explicit obstacle information: if the robot hand touches any non-target object, we will give a large penalty (e.g. -10). We use the same experimental setting in the single-object grasping tasks, including training iterations, and evaluation criteria.

2) Results: Table II shows the quantitative results, demonstrating that our method consistently outperforms all baseline methods. Unlike baselines, our approach has **no** noticeable performance degradation while our baseline **fails in almost all** the scenarios. By leveraging high-level trajectory guidance without requiring additional penalty terms, our method demonstrates strong planning and search capabilities, enabling rapid adaptation to novel, cluttered environments.

SR(%) ↑ Ref. Trajectory	Cluttered scene-1 `mustard`	`bleach`	Cluttered scene-2 `bleach`	`sugar`	Cluttered scene-3 `bleach`	`sugar`	Avg.
None	7	12	0	3	0	0	4
Human-Motion-Retargeting	0	0	0	0	0	0	0
Trajectory Generator	0	0	18	2	0	0	3
Ours	**77**	**85**	**82**	**83**	**78**	**62**	**78**

TABLE II

RESULTS IN SIMULATION ON THREE CLUTTERED ENVIRONMENTS.

C. Real World Experiments

1) Experiment Setup: **Task Setup.** We validate the zero-shot transfer of our policies on a real robot across six environments, including single-object grasping and cluttered scenarios. For *seen* and *unseen* single-object grasping tasks, we apply the same randomization range: ± 5 cm in the xy plane and $\pm 30^\circ$ rotation around the z-axis. For cluttered environments, we use a variation of ± 3 cm in xy and $\pm 10^\circ$ in z rotation. We select the best-performing seeds for evaluation and repeat each experiment ten times. Please see Hardware Setup and Implementation Details in appendix A-B.

SR (%) ↑ Ref. Trajectory	Single-seen `mustard`	Single-unseen `bleach`	Cluttered scene-1 `mustard`	`bleach`	Cluttered scene-2 `bleach`	Cluttered scene-3 `bleach`	Avg.
None	20	50	0	0	0	0	13
Human-Motion-Retargeting	50	80	0	0	0	0	22
Trajectory Generator	60	60	0	0	0	0	20
Ours	**80**	**90**	**80**	**80**	**70**	**70**	**78**

TABLE III

REAL-WORLD EXPERIMENTS ON SIX ENVIRONMENTS, INCLUDING *seen*, *unseen*, AND CLUTTERED SCENARIOS.

2) Results: Shown in Table III, our policy transfers to the real world with minimum performance degradation and significantly outperforms all baseline methods. Notably, while for *seen*-`mustard` task, the Human-Retargeting baseline performs well in simulation, its grasp posture fails to ensure a stable grasp in the real world, leading to execution failures. Interestingly, for the *unseen*-`bleach` task, both RL and S2S baselines achieve higher success rates than in simulation, primarily because we selected a successful seed for deployment. All baselines fail in cluttered environments, which aligns with their near-zero success rates observed in simulation. Figure 4 provides qualitative examples of robot using our methods to successfully grasp objects while navigating around obstacles.

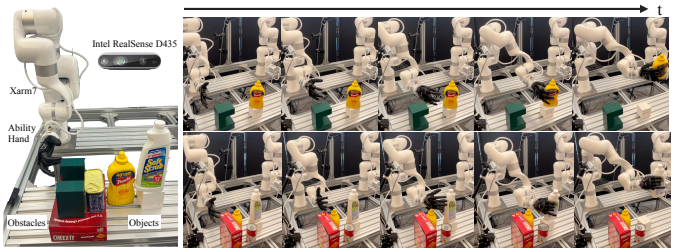


Fig. 4. **Real-World Experiments**. Left: experimental setup. Right: examples of the robot successfully grasping objects while navigating around obstacles.

IV. CONCLUSION

We propose to learn a particle-based world model from human to facilitate robot learning. We use the model to adaptively search for diverse trajectories for different object configurations. We successfully demonstrate the effectiveness of our methods under various settings in simulation and real-world robots.

References

[1] Bo Ai, Stephen Tian, Haochen Shi, Yixuan Wang, Cheston Tan, Yunzhu Li, and Jiajun Wu. Robopack: Learning tactile-informed dynamics models for dense packing. *arXiv preprint arXiv:2407.01418*, 2024.

[2] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[3] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. $\pi 0$: A vision-language-action flow model for general robot control, 2024. *URL https://arxiv. org/abs/2410.24164*.

[4] Justin Carpentier, Guilhem Saurel, Gabriele Buondonno, Joseph Mirabel, Florent Lamiraux, Olivier Stasse, and Nicolas Mansard. The pinocchio c++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives. In *IEEE International Symposium on System Integrations (SII)*, 2019.

[5] Yu-Wei Chao, Wei Yang, Yu Xiang, Pavlo Molchanov, Ankur Handa, Jonathan Tremblay, Yashraj S Narang, Karl Van Wyk, Umar Iqbal, Stan Birchfield, et al. Dexycb: A benchmark for capturing hand grasping of objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9044–9053, 2021.

[6] Jiayi Chen, Mi Yan, Jiazhao Zhang, Yinzhen Xu, Xiaolong Li, Yijia Weng, Li Yi, Shuran Song, and He Wang. Tracking and reconstructing hand object interactions from point cloud sequences in the wild. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 304–312, 2023.

[7] Tao Chen, Megha Tippur, Siyang Wu, Vikash Kumar, Edward Adelson, and Pulkit Agrawal. Visual dexterity: In-hand reorientation of novel and complex object shapes. *Science Robotics*, 8(84):eadc9244, 2023.

[8] Yuanpei Chen, Chen Wang, Yaodong Yang, and C Karen Liu. Object-centric dexterous manipulation from human motion data. *arXiv preprint arXiv:2411.04005*, 2024.

[9] Zerui Chen, Shizhe Chen, Arlaud Etienne, Ivan Laptev, and Cordelia Schmid. ViViDex: Learning vision-based dexterous manipulation from human videos. 2024.

[10] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.

[11] Abhishek Gupta, Clemens Eppner, Sergey Levine, and Pieter Abbeel. Learning dexterous manipulation for a soft robotic hand from human demonstrations. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3786–3793. IEEE, 2016.

[12] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[13] Yana Hasson, Gul Varol, Dimitrios Tzionas, Igor Kalevatykh, Michael J Black, Ivan Laptev, and Cordelia Schmid. Learning joint reconstruction of hands and manipulated objects. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11807–11816, 2019.

[14] Zhengdong Hong, Kangfu Zheng, and Linghao Chen. Fully automatic hand-eye calibration with pretrained image models. *International Conference on Intelligent Robots and Systems (IROS)*, 2024.

[15] Wanxin Jin. Complementarity-free multi-contact modeling and optimization for dexterous manipulation. *arXiv preprint arXiv:2408.07855*, 2024.

[16] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

[17] Yunzhu Li, Jiajun Wu, Russ Tedrake, Joshua B Tenenbaum, and Antonio Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. *arXiv preprint arXiv:1810.01566*, 2018.

[18] Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training. *arXiv preprint arXiv:2210.00030*, 2022.

[19] Russell Mendonca, Shikhar Bahl, and Deepak Pathak. Structured world models from human videos. 2023.

[20] Anusha Nagabandi, Kurt Konolige, Sergey Levine, and Vikash Kumar. Deep dynamics models for learning dexterous manipulation. In *Conference on robot learning*, pages 1101–1112. PMLR, 2020.

[21] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.

[22] Tao Pang, HJ Terry Suh, Lujie Yang, and Russ Tedrake. Global planning for contact-rich manipulation via local smoothing of quasi-dynamic contact models. *IEEE Transactions on robotics*, 39(6):4691–4711, 2023.

[23] Yuzhe Qin, Yueh-Hua Wu, Shaowei Liu, Hanwen Jiang, Ruihan Yang, Yang Fu, and Xiaolong Wang. Dexmv: Imitation learning for dexterous manipulation from human videos. In *European Conference on Computer Vision*, pages 570–587. Springer, 2022.

[24] Yuzhe Qin, Wei Yang, Binghao Huang, Karl Van Wyk, Hao Su, Xiaolong Wang, Yu-Wei Chao, and Dieter Fox. Anyteleop: A general vision-based dexterous robot arm-hand teleoperation system. *arXiv preprint*

arXiv:2307.04577, 2023.

[25] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

[26] Javier Romero, Dimitrios Tzionas, and Michael J Black. Embodied hands: Modeling and capturing hands and bodies together. *arXiv preprint arXiv:2201.02610*, 2022.

[27] Himanshu Gaurav Singh, Antonio Loquercio, Carmelo Sferrazza, Jane Wu, Haozhi Qi, Pieter Abbeel, and Jitendra Malik. Hand-object interaction pretraining from videos. *arXiv preprint arXiv:2409.08273*, 2024.

[28] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[29] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

[30] 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.

[31] 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.

[32] 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.

[33] Grady Williams, Paul Drews, Brian Goldfain, James M Rehg, and Evangelos A Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 1433–1440. IEEE, 2016.

[34] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, et al. Sapien: A simulated part-based interactive environment. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11097–11107, 2020.

Supplementary Materials for Submission ID-11

APPENDIX A
IMPLEMENTATION DETAILS

A. Simulation and Policy Implementation Details

Simulation environment. We use a PSYONIC Ability Hand mounted on an xArm 7 to match our real-world hardware setup. For simulation, we employ SAPIEN [34] and OpenAI-Gymnasium [29] to enable multi-process parallel simulation.

Implementation. We use hand-object interaction data from DexYCB, filter out *unseen* objects, extract sequences of hand and object particles. We represent the hand with 40 particles and each object with 100 particles. For sampling, we generate 1000 samples per iteration and update over 15 iterations which takes around 60 seconds. Policy learning is performed using Soft Actor-Critic(SAC)[12], and we set the maximum environmental steps at 7.68 million steps for training which is enough for all the baselines to converge. We apply domain randomization and object initial configuration randomization with an xy translation range of ± 5 cm and a rotation range of $\pm 30^\circ$ around the z-axis. A grasp is considered successful if the object is lifted by at least 8 cm.

B. Real-World Experiments

Hardware setup. We use a PSYONIC Ability Hand mounted on an xArm 7 for real-world experiments, as illustrated in Figure 4. A RealSense D435i depth camera, calibrated following [14], is used to capture RGB-D data for object pose estimation.

Implementation. We use FoundationPose [32] for object pose estimation, which tracks the object pose given object meshes, RGB-D observations, and an initial object segmentation mask. At the start of each experiment, we manually select a bounding box around the target object in the input image and apply Segment Anything (SAM) [16] to extract the initial segmentation mask. Using the estimated pose, we align the simulation by initializing the object to match its real-world configuration. The policy is then executed in simulation for 120 timesteps, producing a sequence of joint positions that are used as reference targets for execution on the real robot.

C. Reinforcement Learning

In the RL reward function (equation 3) in the main paper, there are 4 items. The first term, following reward R_f, defined as $R_f = \beta_1 R_j + \beta_2 R_{\text{ee},x} + \beta_3 R_{\text{ee},r}$, encourages the robot to follow reference hand motions. Here, the first component $R_j = 1 - \tanh(\alpha_1 \|q^t - \hat{q}^{t'}\|)$ encourage the robot hand to follow reference joint positions, where q_t and $\hat{q}_{t'}$ are robot hand joint positions at timestep t and reference robot hand joint positions at corresponding timestep t'. The second component $R_{\text{ee},x} = 1 - \tanh(\alpha_2 \|x^t_{\text{ee}} - \hat{x}^{t'}_{\text{ee}}\|)$ and the third component $R_{\text{ee},r} = 1 - \tanh(\alpha_3 \phi(\theta^t, \hat{\theta}^{t'}))$ constrains robot ee pose where x_{ee} are robot ee position and ϕ computes the angular distance between robot ee orientation θ^t with its reference $\hat{\theta}^{t'}$.

We use a small weight on the joint following reward R_j, as reference motions can be imprecise. Instead, we encourage finger exploration through R_{contact} , which gives a reward only when the thumb and at least one other finger make contact with the object. We assign an additional task-success related lifting reward $R_{\text{lifting}} = \lambda_5 h$ where h is the lifted height (cm) compared to the initial pose of the object, and give a large reward R_{success} when task success. In experiments, we **do not** tune the parameters for the reinforcement learning policy for different object or different task. By contrast, we use the same set of the parameters for all the experiments shown below: $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_5 = 1, \lambda_4 = 15, \beta_1 = 0.1, \beta_2 = \beta_3 = 1, \alpha_1 = \alpha_2 = \alpha_3 = 1$

APPENDIX B
ADDITIONAL EXPERIMENTS

A. Generalization to Diverse Object Poses

	pose-1	pose-2	pose-3	pose-4	pose-5	pose-6	Avg.
Success Rate (%)	83	76	75	69	72	74	75

TABLE IV
SUCCESS RATES OF MUSTARD BOTTLE UNDER 6 NOVEL POSES

In Table IV, we conduct experiments for testing the generalizability of our methods under the various object poses. We **randomly** selected 6 distinct poses (which is unseen during the world model training) in a area with the size of a US-letter paper. Results demonstrates that our method could adapt to novel object positions and orientations well. Results can also be found in video materials.

APPENDIX C
DISCUSSION ON DYNAMICS MODEL IMPLEMENTATION

A. Graph Construction Process

A key requirement for our dynamics model training is maintaining consistent particle correspondences over time, i.e. each particle remains in the same region of hand or object throughout the entire trajectory. We adopt tailored strategies for sampling and processing points on the object, human, and robot hand as illustrated in Figure 5.

Object Point Sampling Our method assume access to the object mesh. A fixed set of surface points is obtained by uniformly sampling the mesh. These points are transformed using the object's pose at each frame, resulting in a consistent and trackable point cloud across time.

Human Human (MANO) Sampling Human hand Human hand pose is represented using the MANO model [26], which deforms a shared mesh template over time. We apply furthest point sampling (FPS) to a *flat hand pose* (canonical pose) to select a fixed subset of mesh vertices, which are then extracted from the MANO mesh at each frame to ensure one-to-one temporal correspondence.

Robot Hand Sampling We adopt a *particle-level forward kinematics* approach to generate pose-aware particle sets consistent with the robot's motion. For each hand link, a dense

point cloud is sampled in its canonical (link-local) pose, and each point is associated with its link. At each frame, we apply link transformations via forward kinematics to obtain consistent particle positions. To reduce redundancy, furthest point sampling is applied on a *flat robot hand pose*.

Edge Construction We construct intra-hand edges (human or robot) by connecting particle pairs within 0.04 m in the *flat hand pose*. Intra-object edges use 5-nearest neighbors (5-NN) in the canonical pose. Hand-object edges are built dynamically per frame between hand and object particles within 0.04 m.

B. Implementation Details

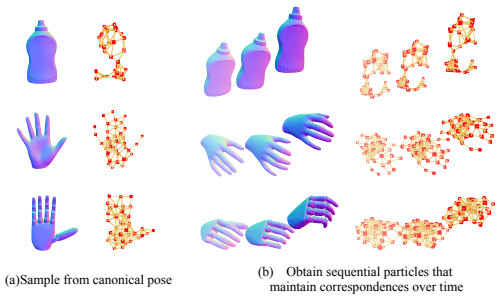
The dynamics model is trained with MSE loss using Adam Optimzier with a learning rate 5×10^{-4} and batchsize 32. Hyperparameters for dynamics models are shown in Table V. Following [1], we restrict the magnitude of the rotation component of predicted rigid transformations for a single step to be at most 30 degrees to stabilize training, which is much larger than any rotation that occurs in our datasets. Model training converges within 1 hours with one NVIDIA RTX 4090 GPU.

Hyperparameters	Value	Hyperparameters	Value
Training sequence length	5 steps	Testing sequence length	30 steps
# graph points per object	40	# graph points per hand	40
Node encoder MLP width	150	Edge encoder MLP width	150
Node encoder MLP layers	3	Edge encoder MLP layers	3
Edge effect MLP width	150	Edge effect MLP layers	3
Edge propagation steps	3		

TABLE V
HYPERPARAMETERS FOR DYNAMICS MODEL.

C. Ablation on Graph Construction Choice

We perform ablation study on graph construction choice using DexYCB [5] dataset, following its default train and test split. We compare our model with a variant where edge connectivity is independently re-computed at each time step without preserving temporal consistency.



(a)Sample from canonical pose (b) Obtain sequential particles that maintain correspondences over time

Fig. 5. **Graph Construction Process** for Dynamics Model. (a) Hand and object particles, along with intra-hand and intra-object edges, are sampled from a canonical pose. (b) These particles maintain consistent correspondences across the sequence, resulting in temporally coherent graphs.

We evaluate model performance using three metrics: Chamfer Distance, Earth Mover's Distance (EMD), and translation difference (t_{diff}) between the predicted and ground truth object states. As shown in Table VI, our full model achieves lower errors across all metrics, demonstrating the benefit of temporally consistent graph structure. This consistency acts as an inductive prior, encouraging the model to focus more effectively on capturing hand-object interactions, leading to improved dynamics prediction.

	Chamfer ↓	EMD ↓	t_{diff} ↓
Ours	**0.0179**	**0.0140**	**0.0161**
Ours w/o Edge consistency	0.0180	0.0146	0.0165

TABLE VI
ABLATION STUDY ON GRAPH CONSTRUCTION. WE COMPARE OUR FULL MODEL WITH A VARIANT WITHOUT TEMPORAL EDGE CONSISTENCY, EVALUATED USING CHAMFER DISTANCE, EMD, AND t_{DIFF} OF PREDICTED OBJECT STATES WITH GROUND-TRUTH STATES.

APPENDIX D
IMPLEMENTATION DETAILS IN MPC

A. Cost Function Definitions

We use a combination of cost functions as planning objectives.

Point Cloud Distance Cost penalizes the distance between predicted object particles to the goal object particles. Decay factor γ reduces the influence of earlier time steps, placing more emphasis on aligning the predicted object with the goal configuration in the final frame, which is the most important.

$$\mathcal{L}_{\text{pc}} = \sum_{t=0}^{H-1} \gamma^{H-t-1} \|\hat{x}_t^{\text{obj}} - x_{\text{goal}}^{\text{obj}}\|_2 \tag{4}$$

Finger-to-Object Distance Cost and **Palm-to-Object Distance Cost** penalize the distance between the hand (specifically, the fingers and palm) and the object, thereby encouraging reaching behaviors. At each timestep, the minimum distance between any finger (or palm) particle and any object particle is computed and summed across the prediction horizon.

$$\mathcal{L}_{\text{finger}} = \sum_{t=0}^{H-1} \min_{i,j} \|\hat{x}_{i,t}^{\text{finger}} - \hat{x}_{j,t}^{\text{obj}}\|_2 \tag{5}$$

$$\mathcal{L}_{\text{palm}} = \sum_{t=0}^{H-1} \min_{i,j} \|\hat{x}_{i,t}^{\text{palm}} - \hat{x}_{j,t}^{\text{obj}}\|_2 \tag{6}$$

Here, $\hat{x}_{i,t}^{\text{finger}}$ and $\hat{x}_{i,t}^{\text{palm}}$ denote the predicted positions of finger or palm particles at timestep t. These particle groups are predefined on MANO[26] mesh template or robot hand links.

Penetration Cost enforces a key physical constraint to prevent hand-object intersection, inspired from priors work in

hand-object pose estimation[13, 6]. Specifically, it penalizes predicted hand particles that penetrated the object:

$$\mathcal{L}_{\text{pene}} = \sum_{t=0}^{H-1} \sum_{v \in \hat{x}_t^{\text{hand}}} (-\mathbb{1}_{\phi(\hat{P}^{-1}v)<0} \quad \phi(\hat{P}^{-1}v)) \tag{7}$$

where $\phi(x)$ is the trilinear interpolated SDF value at location x from the object's SDF volume. $\hat{P}^{-1}v$ transforms the hand particles back into the object's canonical coordinate space, using a transformation $\hat{P}$ derived from corresponding points alignment between particles in the object's canonical space and those in the predicted state.

Attraction Cost encourages contact between the fingertips and the object [13] by penalizing the minimum SDF values of the five fingerts that are outside the object:

$$\mathcal{L}_{\text{attr}} = \sum_{t=0}^{H-1} \sum_{i=i}^{n=5} \min_{v \in \hat{x}_t^{\text{j-th finger}}} (\mathbb{1}_{\phi(\hat{P}^{-1}v)>0} \quad \phi(\hat{P}^{-1}v)), \tag{8}$$

where $\hat{x}_t^{\text{j-th finger}}$ is predefined j-th finger particles. We follow the same strategy as [6] to determine when the attraction cost should be applied: when the maximum penetration (i.e. the maximum of the negative SDF) exceeds a threshold, the hand is considered to be in contact with the object, and the attraction cost is activated to pull distant fingers closer.

We set $\gamma = 0.8, \omega_1 = 30, \omega_2 = 0.5, \omega_3 = 0.5, \omega_4 = 5, \omega_5 = 0.5$ in all experiments.

B. Details in MPPI Planning

We sample a sequence of keyframe robot commands $\{u_t\}_{t=0}^{H'}$, where each command has $(6+n)$ degrees of freedom (DoF): 3 DoF for wrist position, 3 DoF for wrist orientation represented in axis-angle form, and n DoF for joint positions. These keyframe sequence is then upsampled to the full trajectory length H by applying linear interpolation to the wrist position and joint positions, and spherical linear interpolation (slerp) to the wrist orientation. The hyperparameters used for the MPPI optimizer during planning with the learned dynamics model are summarized in Table VII.

Hyperparameters	Single Object	Cluttered Environment
Action sampler temporal correlation β^*	0.3	0.3
MPPI # samples	1000	1000
Robot commands keyframe horizon H'	9	11
Robot commands full horizon H	30	30
MPPI # iterations	15	30
MPPI scaling temperature γ^*	1	1

TABLE VII
HYPERPARAMETERS FOR PLANNING. WE USE NOTATION FROM [20] FOR PARAMETERS DENOTED BY $*$.