Parameterized Quasi-Physical Simulators for Dexterous Manipulations Transfer
Supplementary Materials

Xueyi Liu[1,3], Kangbo Lyu[1], Jieqiong Zhang[1], Tao Du[1,2,3], and Li Yi[1,2,3]

[1] Tsinghua University [2] Shanghai AI Laboratory [3] Shanghai Qi Zhi Institute
https://meowuu7.github.io/QuasiSim

Overview. The appendix contains a list of supplementary materials to support the main paper.

– **Additional Technical Explanations (Section A)**. We provide additional explanations to complement the main paper.
  - *Dexterous Manipulation Transfer (Section A.1)*. We provide a more formal task definition, outlining its objectives and the involved functions.
  - *Parameterized Quasi-Physical Simulators (Section A.2)*. Detailed explanations of the parameterized point set dynamics, including its full dynamic equations, and the parameterized residual physics, covering network designs, features, input, and output details.
  - *Dexterous Manipulation Transfer via a Physics Curriculum (Section A.3)*. We include comprehensive illustrations of the transfer process based on point sets, the iterative optimization procedure for approximating realistic dynamics, and detailed MPC procedure.

– **Additional Experiments (Section B)**. We present further experimental results to demonstrate the effectiveness of our method, along with discussions, analyses, additional comparisons, a user study and insights into failure cases and limitations.
  - *Transferred Dexterous Manipulations (Section B.1)*. Additional qualitative results showcasing intricate manipulations to highlight our method's capability.
  - *Further Discussions and Analysis (Section B.2)*. We delve deeper into the role of MPC in our method, the question of does the residual physics module really compensates for the estimation other than taking the main role, the intermediate optimization processes in the quasi-physical simulator curriculum, and experiments on a different simulated robot hand whose morphology is significantly different from the human hand to demonstrate our method's capability in such cases.
  - *Additional Comparisons (Section B.3)*. In addition to comparisons with previous Reinforcement Learning (RL) methods, we compare approaches that incorporate human demonstrations into policy learning for acquiring skills.
  - *Failure Cases (Section B.4)*. Analysis of failure cases to gain insights into limitations and areas for improvement.

  - *User Study (Sec. B.5)*. We additionally include a user study to further assess effectiveness of our method.
- **Experimental Details (Section C)**. We illustrate details of datasets, metrics, baselines, models, evaluation settings, and running time as well as the complexity analysis.
- **Potential Negative Societal Impact (Section D).** We discuss the potential negative social impacts of the work.

We include a **website** and a **video** to introduce our work. The website and the video contain *animated transferred dexterous manipulations*. We highly recommend exploring these resources for an intuitive understanding of the task, difficulties, the effectiveness of our model, and its superiority over prior approaches. We include source code in the supplemental material. We will publicly release the code and data upon acceptance of the paper.

A Additional Technical Explanations

We include a figure providing a comprehensive overview of the method (see Fig. 1).

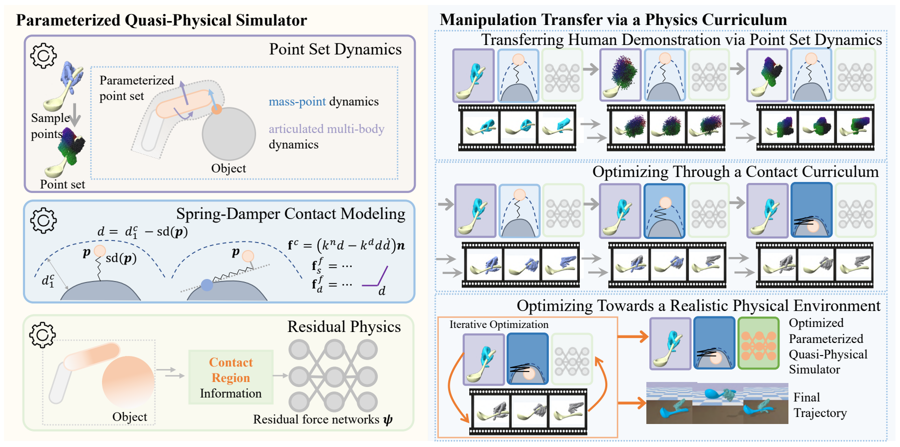

Parameterized Quasi-Physical Simulator
Point Set Dynamics
Parameterized point set
Sample points
Point set
Object
mass-point dynamics
articulated multi-body dynamics
Spring-Damper Contact Modeling
$d = d_i^c - \text{sd}(\mathbf{p})$
$\mathbf{p}$
sd($\mathbf{p}$)
d_i^c
$\mathbf{f}^c = (k^n d - k^d d\dot{d})\mathbf{n}$
$\mathbf{f}_s^f = \cdots$
$\mathbf{f}_d^f = \cdots$
d
Residual Physics
Contact Region Information
Object
Residual force networks $\boldsymbol{\psi}$
Manipulation Transfer via a Physics Curriculum
Transferring Human Demonstration via Point Set Dynamics
Optimizing Through a Contact Curriculum
Optimizing Towards a Realistic Physical Environment
Iterative Optimization
Optimized Parameterized Quasi-Physical Simulator
Final Trajectory

Fig. 1: Detailed Method Overview. The **parameterized quasi-physical simulator** relaxes the articulated multi rigid body dynamics as the *parameterized point set dynamics*, controls the contact behavior via an unconstrained *parameterized spring-damper contact model*, and compensates for unmodeled effects via *parameterized residual physics networks*. We tackle the difficult dexterous manipulation transfer problem via **a physics curriculum**.

A.1 Dexterous Manipulation Transfer

Given a human manipulation demonstration, composed of a human hand mesh trajectory and an object pose trajectory $\{\mathcal{H} = \{\mathbf{H}_n\}_{n=1}^N, \mathcal{O} = \{\mathbf{O}_n\}_{n=1}^N\}$ with N frames, the goal is transferring the demonstration to a dexterous robot hand in simulation. Formally, we aim to optimize a control trajectory $\mathcal{A}$ that drives the dexterous hand to manipulate the object in a realistic simulated environment so that the resulting hand trajectory $\hat{\mathcal{H}} = \{\hat{\mathbf{H}}_n\}_{n=1}^N$ and the object trajectory $\hat{\mathcal{O}} = \{\hat{\mathbf{O}}_n\}_{n=1}^N$ are close to the reference motion $\{\mathcal{H}, \mathcal{O}\}$. Since the object properties and the system parameters are unknown from the kinematics-only trajectory, we estimate such parameters, denoted as set $\mathcal{S}$, along with the hand control optimization.

Optimization objective. The task aims at optimizing a hand control trajectory $\mathcal{A}$ so that the resulting hand trajectory $\hat{\mathcal{H}}$ and the object trajectory $\hat{\mathcal{O}}$ are close to the reference motions $\{\mathcal{H}, \mathcal{O}\}$. Formally, the objective is:

$$\text{minimize}_{\mathcal{A},\mathcal{S}} w^o f^{\mathcal{O}}(\mathcal{O}, \hat{\mathcal{O}}) + w^h f^{\mathcal{H}}(\mathcal{H}, \hat{\mathcal{H}}), \tag{1}$$

where w^o and w^h are object tracking weight and the hand tracking weight respectively, $f^{\mathcal{O}}$ measures the difference between two object pose trajectories, and $f^{\mathcal{H}}$ calculates the difference between two hand trajectory. Specifically,

$$f^{\mathcal{O}}(\mathcal{O}, \hat{\mathcal{O}}) = \frac{1}{N}\sum_{n=1}^{N}((1 - \mathbf{q}_n \cdot \hat{\mathbf{q}}_n) + \|\mathbf{t}_n - \hat{\mathbf{t}}_n\|) \tag{2}$$

$$f^{\mathcal{H}}(\mathcal{H}, \hat{\mathcal{H}}) = \frac{1}{N}\sum_{n=1}^{N}\|\mathbf{P}_n^h - \mathbf{P}_n^r\|, \tag{3}$$

where $\mathbf{q}_n$ is the orientation of the n-th frame reference object pose, represented in quaternion, $\mathbf{t}_n \in \mathbb{R}^3$ is the translation of the n-th frame reference object pose, $\hat{\mathbf{q}}_n$ and $\hat{\mathbf{t}}_n$ are the quaternion and the translation of the n-th frame estimated object pose, $\mathbf{P}_n^h$ is the reference human hand keypoint at the n-th frame, and $\mathbf{P}_n^r$ is the estimated robot hand keypoint at the n-th frame correspondingly. Keypoints consist of five fingertips and three points on the hand wrist. We manually defined them (Fig. 2). Weights w^o and w^h are set to 1.0, 1.0 in our method.

A.2 Parameterized Quasi-Physical Simulators

Parameterized point set dynamics. Each point $\mathbf{p}_i$ in the point set $\mathcal{Q}$ is treated as a mass point with a finite mass $\mathbf{m}_i$ and infinitesimal volume. The action space of the point set is composed of the joint forces $\mathbf{u} \in \mathbb{R}^{n_r}$ in the reduced coordinate system, alongside a 3 degrees of freedom free force $\mathbf{a}_i \in \mathbb{R}^3$ applied to each point $\mathbf{p}_i \in \mathcal{Q}$. A point is considered to be "attached" to the body it was sampled from and can undergo articulated transformations, as illustrated in the example shown in Figure 3. The dynamics of the point set encompass

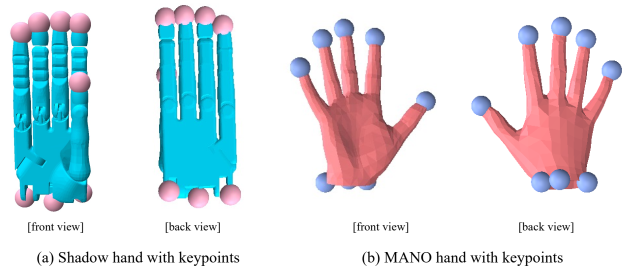


Fig. 2: Hands with keypoints (keypoints are drawn as large pink and blue purple points).

articulated multi-body dynamics [8, 11], along with the mass point dynamics of each individual point $\mathbf{p}_i$. Specifically,

$$\mathbf{M}_r \ddot{\mathbf{q}}_r = \tilde{\mathbf{f}}_r + (1-\alpha)\mathbf{J}_{mr}^T \mathbf{f}_m + \mathbf{f}_{QVV} + \mathbf{u}, \tag{4}$$

$$m_i \ddot{\mathbf{x}}_i = \mathbf{J}_i \mathbf{u} + \alpha \mathbf{f}_i + \alpha, \forall \mathbf{p}_i \in \mathcal{Q}, \tag{5}$$

where $\mathbf{M}_r \in \mathbb{R}^{n_r \times n_r}$ is the generalized inertia matrix in reduced coordinates, n_r is the number of freedom of the articulated object, $\mathbf{q}_r \in \mathbb{R}^{n_r}$ is the reduced state vector of the articulated object, $\tilde{\mathbf{f}}_r$ is the reduced force vector generated by joint-space such as joint damping and stiffness, $\mathbf{J}_{mr}$ is the Jacobian mapping generalized velocity $\dot{\mathbf{q}}_r$ to its maximal coordinate counterpart $\dot{\mathbf{q}}_m$, $\mathbf{f}_m$ is the maximal wrench vector including force and torque generated in maximal coordinate system, $\mathbf{f}_{QVV}$ is the quadratic velocity vector, $\mathbf{u}$ denotes the generalized joint force, $\mathbf{J}_i$ represents the Jacobian mapping from the generalized velocity to the point velocity $\dot{\mathbf{x}}_i$, $\mathbf{f}_i$ accounts for external forces acting on $\mathbf{p}_i$, and $\mathbf{a}_i \in \mathbb{R}^3$ represents the actuation force applied to the point $\mathbf{p}_i$. Consequently, the point set is controlled by a shared control in the reduced coordinate space $\mathbf{u}$ and per-point actuation force $\mathbf{a}_i$.

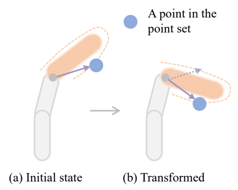


Fig. 3: A point in the point set is regarded as "attached" to the body it sampled from and is affected by joint actions accordingly.

Parameterized residual physics. We introduce two residual contact force networks to compensate for the inherent limitations of the spring-damper based contact modeling. For detailed residual contact force prediction, we introduce a local contact network $f_{\psi_{\text{local}}}$ that utilizes contact information identified in the parameterized contact model and predicts residual forces between each contact pair. For each point pair in contact $(\mathbf{p}, \mathbf{p}^o)$, the local contact region is composed of N_c^l object surface points and N_c^l hand surface points. For the contact point in the object surface $\mathbf{p}^o$, we identify a region which contains object surface points whose distance to point $\mathbf{p}^o$ is not larger than a threshold $d_{thres}^l = 0.05$ (5cm) (point $\mathbf{p}^o$ is not included in the region). After that, $N_c^l - 1$ points are sampled from such points via farthest point sampling. These points, together with $\mathbf{p}^o$ are taken as the object local contact surface points. N_c^l hand points are sampled in the same way. We set N_c^l to 100 in experiments. After that, the local contact information consists of the geometry of the local contact region $\mathbf{P}_c^l \in \mathbb{R}^{2N_c^l \times 3}$, per-point velocity $\mathbf{V}_c^l \in \mathbb{R}^{2N_c^l \times 3}$, and per-object point normal $\mathbf{N}_c^l \in \mathbb{R}^{N_c^l \times 3}$. A PointNet is used to encode the contact region feature. The feature of each point is composed of the point type embedding vector (128 dimension), point position, point velocity, point normal (all zeros for hand points). The hidden dimensions are [128, 256, 512, 1024]. After that, we calculate the global feature via a 'maxpool' operation. Then the global features is fed into the contact force prediction module for local residual contact force prediction. The prediction network is an MLP with hidden dimensions [512, 256, 128]. `ReLU` is leveraged as the activation layer.

To address discrepancies in contact region identification between the parameterized contact model and real contact region, we also incorporate a global residual network $f_{\psi_{\text{global}}}$ that predicts residual forces and torques applied directly to the object's center of mass. To identify a global contact region, we adopt a similar way that first identifies a region on the object, containing object surface points whose distance to the nearest object contact point are smaller than the global contact distance threshold $d_{thres}^g = 0.1$ (10cm). After that, global contact region points are sampled for both the object and the hand in the same way as sampling the local contact region points described above. The number of global contact points on for the object and the hand is $N_c^g = 500$. Subsequently, the global contact region feature is encoded from the global contact region in the same way as does for local contact region feature. Then, the global contact feature is fed to a prediction network for predicting residual force and residual torque. The network architecture is the same as that for local residual force, but with a different output dimension (3 for force, 3 for torque, and 6 dimension in total).

A.3 Dexterous Manipulation Transfer via a Physics Curriculum

Transferring human demonstration via point set dynamics. The articulated rigid constraints are relaxed initially to facilitate robust manipulation transfer between two morphologically different robot hands and to overcome noise in the kinematic trajectory. After we have optimized the control trajectory of the point set constructed from the dynamic MANO hand [5], the next goal is optimizing the control trajectory of the point set constructed from the simulated robot hand. Reliable correspondences between points are required to complete the transfer. Therefore, we first optimize the kinematics-only trajectory of the simulated robot hand based on coarse correspondences defined on keypoints (Fig. 2). The objective is to track the MANO hand trajectory. After that, we define single directional point-point correspondence from the point set of the MANO hand to the point set of the simulated robot hand via the nearest neighbor. That is, for each point in the point set of the MANO hand, we find its nearest point in the point set of the simulated robot hand as its correspondence. After that, the hand tracking objective between the point set of the MANO hand and that of the simulated robot hand becomes the average distance between point-point in correspondence. Subsequently, the control trajectory of the point set is optimized so that the manipulated object pose trajectory can track the reference object pose trajectory, and the trajectory of the simulated robot hand's point set can track the trajectory of the MANO hand's point set. The control trajectory of the point set is first initialized via the kinematic trajectory of the point set via differentiable forward dynamics and optimization.

Optimizing towards a realistic physical environment. When transferring to a realistic physical environment, we iteratively optimize the control trajectory $\mathcal{A}$ and the parameterized simulator. In more detail, in each iteration, the following steps are executed:

- Sample the replay buffer $\mathcal{B}$ from the interested realistic simulated environment.
- Optimize the quasi-physical simulator to approximate realistic dynamics by ensuring that the simulated trajectory closely tracks the trajectory stored in the replay buffer.
- Optimize the control trajectory $\mathcal{A}$ to accomplish the manipulation task within the quasi-physical simulator.

Tracking via closed-loop MPC. After completing the optimization, the final control trajectory is yielded by model predictive control (MPC) [9] based on the optimized parameterized simulator and the hand trajectory $\mathcal{A}$. Specifically, in each step n, the current $\mathbf{A}_n$ and the following controls in several subsequent frames $\{\mathbf{A}_{n+1}, ..., \mathbf{A}_{n+q-1}\}$ are optimized to reduce the tracking error. Denote the simulated object pose trajectory as $\hat{\mathcal{O}}_n^q = \{\hat{\mathbf{O}}_{n+1}, ..., \hat{\mathbf{O}}_{n+q}\}$, the corresponding reference object pose trajectory as $\mathcal{O}_n^q = \{\mathbf{O}_{n+1}, ..., \mathbf{O}_{n+q}\}$, the simulated hand trajectory as $\hat{\mathcal{H}}_n^q = \{\hat{\mathbf{H}}_{n+1}, ..., \hat{\mathbf{H}}_{n+q}\}$ with the corresponding keypoint trajectory $\{\mathbf{P}^r_{n+1}, ..., \mathbf{P}^r_{n+1}\}$ and reference hand keypoint trajectory $\{\mathbf{P}^h_{n+1}, ..., \mathbf{P}^h_{n+1}\}$ the objective at each step n is as follows:

$$\text{minimize}_{\mathcal{A}} w^o f^{\mathcal{O}}(\hat{\mathcal{O}}_n^q, \mathcal{O}_n^q) + w^h f^{\mathcal{H}}(\{\mathbf{P}^r_{n+1}, ..., \mathbf{P}^r_{n+1}\}, \{\mathbf{P}^h_{n+1}, ..., \mathbf{P}^h_{n+1}\}). \quad (6)$$

We update the control trajectory to minimize the objective via 10 steps gradient descent with a learning rate 10^{-4}.

B Additional Experiments

In this section, we present additional experimental results that delve into more qualitative results on challenging cases (see Section B.1), further analysis and discussions (see Section B.2), additional comparisons (see Section B.3), failure case analysis (see Section B.4), and a user study (see Section B.5). Initially, we present additional experimental results achieved by our approach to further demonstrate its effectiveness. Subsequently, we delve into further discussions, including the role of MPC in our method, further investigations in the residual physics module, the intermediate optimization processes in the quasi-physical simulator curriculum, and experiments conducted on a different simulated robot hand that suffers from a significant morphology difference from the human hand. Then we present additional comparisons to the literature where human demonstrations are incorporated into policy learning. After that, we discuss failure cases and analyze our limitations. At last, we present a toy user study as an additional evaluation.

B.1 Transferred Dexterous Manipulations

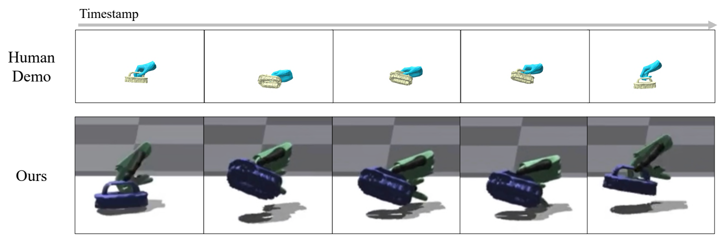
Figure 4 showcases supplementary experimental results obtained through our method. We highly encourage readers to explore our **website** and view the accompanying supplementary video for animated demonstrations.

B.2 Further Discussions and Analysis

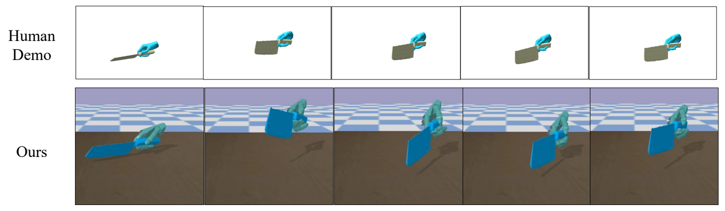
Robustness of MPC. Fig. 5 shows an example demonstrating tracking robustness. In this challenging example where rich contacts between fingers and the palm with the mouse are frequently established and broken, the control sequence optimized in an open-loop manner struggles with keep contacting the mouse, and the tracking is lost finally. However, with the optimized model, the trajectories produced by MPC can successfully maintain enough contact with the object and track the sequence naturally.

Role of residual physics in quasi-physical simulators. We evaluate the role of the residual physics on a small subset of our data from the GRAB dataset. We assess the impact of residual physics on a limited subset of our data from the GRAB dataset. This subset comprises 60 ten-step transitions involving manipulation sequences with objects such as `bunny`, `mouse`, `stapler`, `pyramid`, `cylinder`, `flashlight`, `watch`, `waterbottle`, `hammer`, and `clockarlam`.

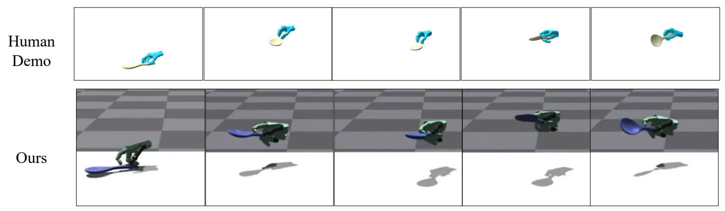
To investigate whether the residual physics compensates for prediction while the analytical simulation remains predominant, we utilize two types of models: one comprising only the analytical part, and the other incorporating both the analytical part and the residual physics network. These models are tasked with predicting the object rotation and translation for each ten-step transition based on the object's initial state and hand action sequence. Let $\mathbf{R}$ denote the object rotation predicted by the analytical part, and $\mathbf{R}_{\text{tot}}$ represent the rotation predicted by the analytical part with the residual model. Similarly, let $\mathbf{t}$



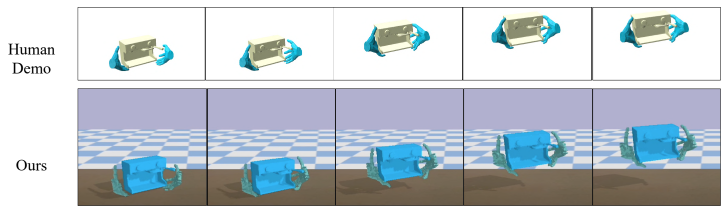
(a) Cleaning things by a brush



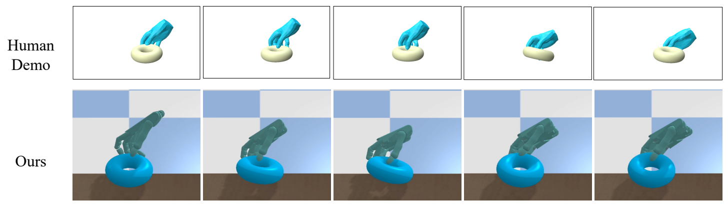
(b) Chopping things using a knife



(c) Carrying things using a ladle



(d) Bimanual cooperation for lifting



(e) Rotating the object up and down

Fig. 4: Transferred manipulations. We provide additional examples to demonstrate the effectiveness of our method. Our approach successfully tracks complex manipulations involving subtle object movements, such as gently shaking a brush for cleaning (*Fig. (a)*), employing non-trivial functional tools (*Fig. (b) (c) (e)*), and executing bimanual cooperation tasks (*Fig. (d)*). For animated demonstrations, please visit **our website** and refer to the accompanying video.

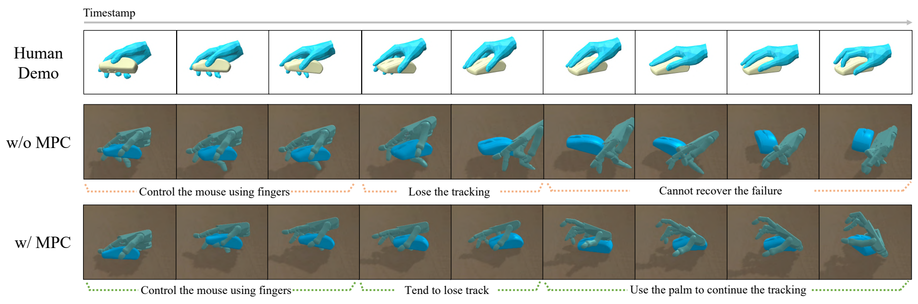

Timestamp
Human Demo
w/o MPC
Control the mouse using fingers
Lose the tracking
Cannot recover the failure
w/ MPC
Control the mouse using fingers
Tend to lose track
Use the palm to continue the tracking

Fig. 5: Robustness of MPC. MPC tries to track the object even after experiencing a dangerous period with the tendency to lose track. While the trajectory yielded by open-loop optimization fails.

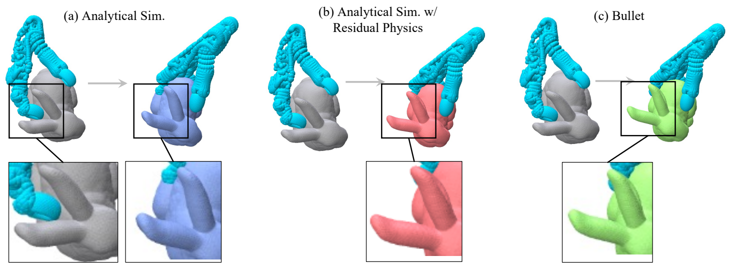


Fig. 6: Analysis on the residual physics module. In this 10-step transition, the transformed bunny predicted by the analytical part of the quasi-physical simulator (purple bunny) only is already close to the GT one (green bunny). The residual physics can compensate for some unmodeled effects. Hence the result (red bunny) yielded by the quasi-physical simulator with both the analytical part and the residual physics module gets closer to the observation in Bullet.

and $\mathbf{t}_{\text{tot}}$ denote the object translation predicted by the analytical part and the analytical part with the residual model, respectively. Therefore, the residual rotation is calculated as $\mathbf{R}_{\text{res}} = \mathbf{R}_{\text{tot}}\mathbf{R}^T$, and the residual translation is calculated as $\mathbf{t}_{\text{res}} = \mathbf{t}_{\text{tot}} - \mathbf{R}_{\text{res}}\mathbf{t}$. Let $\mathbf{V}_{\text{init}}$ denote the initial object vertices, $\mathbf{V}$ represent the transformed vertices, and $\mathbf{V}_{\text{tot}}$ denote the transformed vertices predicted by the analytical part with the residual model. The average per-vertex position difference from the transformed object to the initial object is calculated as

$$p_{\text{diff}} = \frac{1}{N_v}\|\mathbf{V} - \mathbf{V}_{\text{init}}\|. \tag{7}$$

Similarly, the average per-vertex position difference from the transformed object predicted by the total model to the initial object is computed as

$$p_{\text{diff}}^{\text{tot}} = \frac{1}{N_v}\|\mathbf{V}_{\text{tot}} - \mathbf{V}_{\text{init}}\|. \tag{8}$$

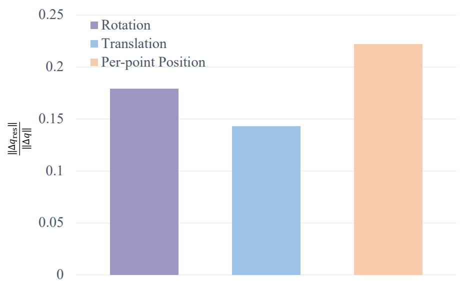


Fig. 7: Quantitative analysis on the residual physics module.

Finally, the average per-vertex position difference from the transformed object predicted by the total model to the object predicted by the analytical part is calculated as:

$$p_{\text{diff}}^{\text{res}} = \frac{1}{N_v} \|\mathbf{V}_{\text{tot}} - \mathbf{V}\|. \tag{9}$$

For each 10-step transition, we calculate the relative quantities of the three types predicted by the residual physics, including the object rotation (measured by angles) $\frac{\text{angle}(\mathbf{R}_{\text{res}})}{\text{angle}(\mathbf{R}_{\text{tot}})}$, object translation $\frac{\mathbf{t}_{\text{res}}}{\mathbf{t}_{\text{tot}}}$, and the object per-point difference $\frac{p_{\text{diff}}^{\text{res}}}{p_{\text{diff}}}$, compared to the overall predicted values by the quasi-physical simulator.

As depicted in the bar chart shown in Figure 6, it is evident that the analytical model plays the primary role in predicting state transitions, while the information predicted by the residual module compensates for the prediction. A visual example is depicted in Figure 6. The `bunny` undergoes rotation by a certain angle in the 10-step transition. The predicted result by the analytical part only is close to the ground-truth transformed object already. This alignment can be readily observed by examining the angle between the two ears of the bunny and the vertical/horizontal line, respectively. The residual physics compensate for unmodeled effects. Hence the object predicted by the full model is closer to the ground-truth transition observed in Bullet.

The optimization process in the quasi-physical simulator curriculum. The quasi-physical simulator curriculum initially relaxes various constraints within the simulator to alleviate the optimization problem. Subsequently, the physics constraints are gradually tightened to enable the optimization to converge towards a solution in a more realistic physics model. Fig. 8 illustrates the intermediate optimization process.

During the first optimization iteration, articulated rigid constraints are relaxed, and the articulated rigid dexterous hand is represented and driven as a point set. Then, articulated constraints are imposed. The optimization continues in the simulator with an increasing contact stiffness (the following three lines in Fig. 8).

Since the articulated dexterous hand is initially represented as a point set, comprised of points sampled from the ambient space of the surface mesh, con-

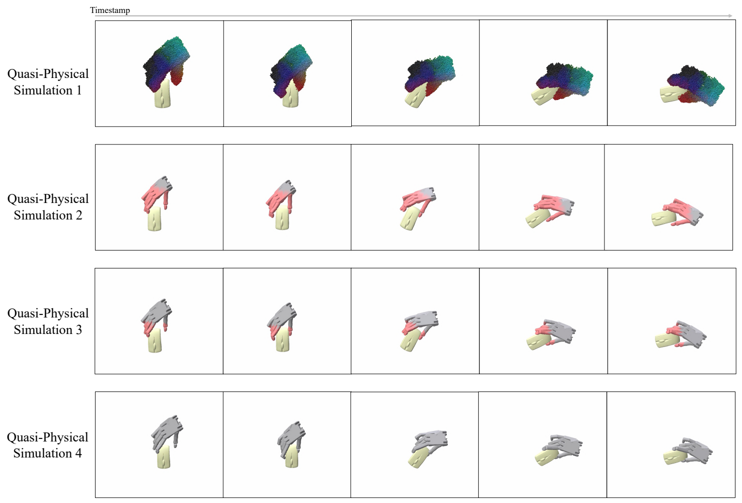
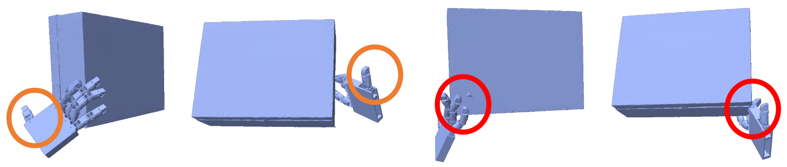

Timestamp
Quasi-Physical Simulation 1
Quasi-Physical Simulation 2
Quasi-Physical Simulation 3
Quasi-Physical Simulation 4

Fig. 8: Example of the optimization process in the **quasi-physical simulator curriculum**. Initially, both the contact constraints and the articulated rigid constraints are relaxed and the object is represented as a point set (the first line). Then the articulated rigid constraints are imposed and the contact model is gradually tightened. The optimization is solved in each of the simulators in the curriculum. We use orange red color to represent the "activated manipulators".

tact between the hand and the object may not necessarily be established immediately. This is because contact can occur between points that are distant from each other, and these points can still act as manipulators. However, even with the articulated constraints removed during the initial optimization stages, the optimization process can still be effectively solved due to the softness of the contact model at the beginning.

As the optimization progresses, we gradually transition towards the final quasi-physical simulator with articulated rigid constraints and the stiffest contact model. In Fig. 8, we use orange red color to represent the "activated manipulators" — surface points where contact can be established between them and the object.

Transferred to a robot hand with a significant morphological difference from the human hand. Utilizing the point set representation, we can facilitate the transfer of manipulation skills to a morphologically different hand. We conducted additional experiments aimed at transferring manipulation from a human hand to a morphologically different robot hand obtained from Diff-Hand [20]. As shown in Fig. 11 (c), the thumb of the dexterous hand is obviously shorter than the human hand. Intuitively, completing manipulations using this hand is difficult. Directly transferring the manipulation via sparse correspondences defined between such two hands (*e.g.,* finger and wrist correspondences as we have defined between the Shadow hand and the human hand (Fig. 2)) is not sufficient, leading to missing contacts and unwanted penetrations shown in



(a) Missing contacts (b) Penetrations and implausible poses

Fig. 9: Functionally implausible transferred poses via sparse correspondences defined by keypoints.

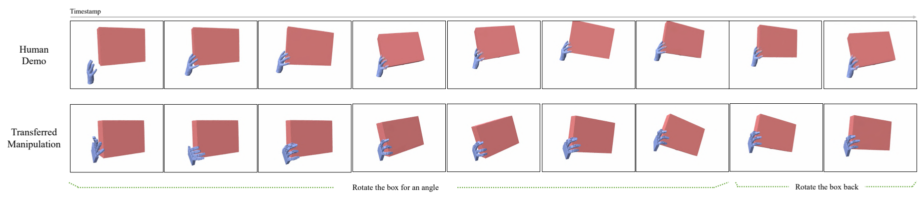


Fig. 10: Manipulations transferred to a morphologically different dexterous robot hand. Taking advantage of the point set representation, the manipulation can be easily transferred to a dexterous hand with an extremely short thumb, which is different from the original MANO hand.

Fig. 9. However, as shown in Fig. 10, our method can still effectively control it to complete the box rotation manipulation. Experiments are conducted in the last quasi-physical simulator from the curriculum.

B.3 Additional Comparisons

The main paper includes comparisons with both model-based and model-free approaches for solving the manipulation transfer task. For model-free methods, we compare with the DGrasp series models. The DGrasp series employs a carefully designed RL-based method for grasping, incorporating well-devised rewards containing position-to-goal information and contact information. Notably, DGrasp's methodology serves as the foundation for their recent work, ArtiGrasp [22]. ArtiGrasp extends the manipulation capabilities to articulated objects and introduces learning techniques such as a gravity curriculum to handle complex relocate-and-articulate task settings. Given the meticulous reward design, stage-wise learning approach, and subsequent improvements, we consider DGrasp as a robust RL-based baseline. However, DGrasp is not explicitly designed for the tracking task, as it relies solely on sparse reference frames obtained from human demonstrations. Therefore, we introduce the improved version of DGrasp-Tracking as our baseline.

Many works have explored the combination of RL and imitation learning to leverage human demonstrations for learning robotic manipulation skills [2–4, 10, 14–16]. In these approaches, human demonstrations are utilized either as

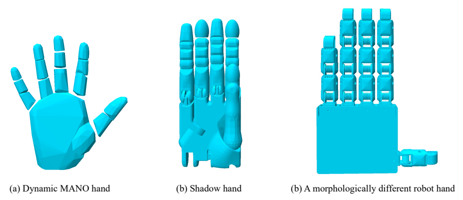


Fig. 11: Comparisons between the dynamic MANO hand (*Fig. (a)*) and two simulated robot hands (*Fig. (b) (c)*) we considered in this work. Compared to the hand shown in Fig. (c), the Shadow hand is more similar to the human hand, but still with morphology differences that cannot be ignored. For fine-grained manipulation tasks, such morphological difference poses significant challenges for transferring. The hand in Fig. (c) is featured by its extremely short thumb and four other fingers longer than the human hand. Transferring human demonstrations to this hand is therefore very difficult. Our flexible point set representation, however, can still work in this case.

Table 1: Additional Comparisons. Quantitative comparisons between our method and DexMV. Experiments are conducted on sequences from the GRAB dataset in the Bullet simulator. **Bold red** numbers for best values.

	Object		Hand		Overall
Method	R_{err} $(^\circ, \downarrow)$	T_{err} $(cm, \downarrow)$	MPJPE $(mm, \downarrow)$	CD $(mm, \downarrow)$	Success Rate $(\%, \uparrow)$
DexMV	28.36	2.42	41.53	18.09	11.11/18.52/48.15
Ours	**22.38**	**1.76**	**35.02**	**13.62**	**25.93**/**37.04**/**62.96**

dense information for the robot to imitate or as sparse reward signals, such as grasp affordances [2]. However, these methods often struggle with the imbalance between human-likeness and task completion, leading to biases towards RL-preferred trajectories.

For the sake of experimental completeness and to showcase the effectiveness of our strategy in contrast to this trend, we compare our approach with DexMV [15]. Among its follow-ups and related works [1, 2, 14], DexMV shares the most similar setting to ours. In DexMV, human demonstrations provide dense references to shape the reward space for their RL algorithm. Furthermore, DexMV is openly available, making it conducive for comparative evaluation[1].

We compare our method with DexMV (DAPG) on a subset, containing manipulation sequences from the GRAB dataset, in the Bullet simulator. Table 1 presents the average quantitative results over the tested sequences. Fig. 12 further leverages some examples to give an intuitive evaluation. In the challenging example shown in Fig. 12 (a) with rich and changing contacts, our method can perform well. However, DexMv struggles to give satisfactory results. In the example shown in Fig. 12 (b), we can track the object in a human-like way. However,

[1] DexMV's GitHub Repository Link

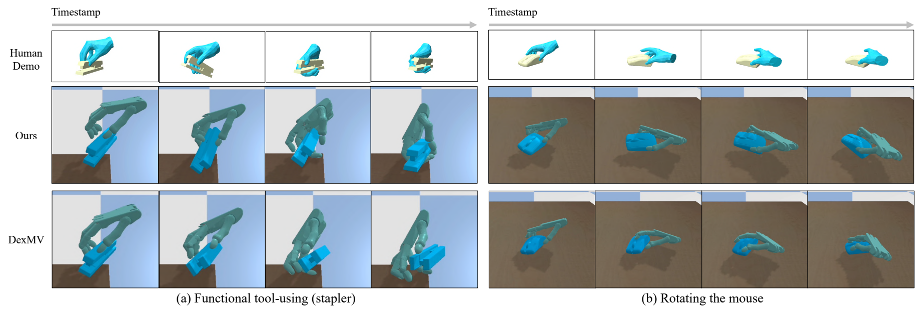


Fig. 12: Visual comparisons between our method and DexMV. We can complete the tracking in a human-like way. However, DexMV cannot fulfill this vision. Its resulting trajectory may deviate from the human demonstration obviously, as observed in both Fig. (a) and (b). Besides, it struggles with the challenging example shown in Fig. (a) with rich and changing contacts.

though DexMV can complete the object tracking task to some extent, the resulting hand trajectory significantly deviates from the human hand demonstration.

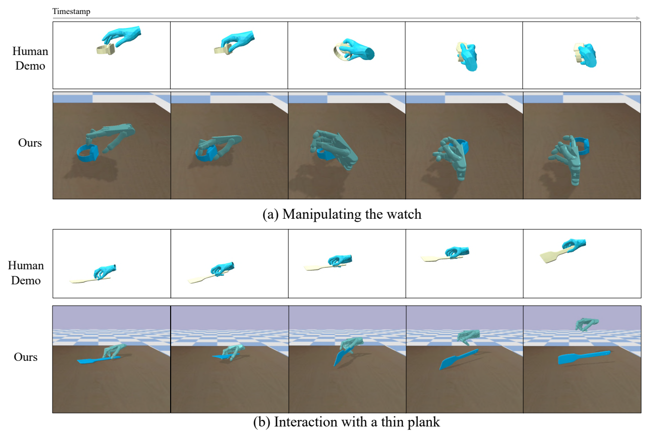


Fig. 13: Failure cases analysis. *Fig. (a)*: The hand fails to grasp the wristwatch, which requires us to control several fingers to pass through the ring of the wristwatch. *Fig. (b)*: The hand fails to find a good strategy for lifting the thin plank.

B.4 Failure Cases

In this section, we delve into the failure cases encountered by our method despite its effectiveness on many sequences. Our method may falter in controlling a simulated robot hand to track manipulation demonstrations in the following scenarios:

- Manipulations requiring highly precise control, such as threading fingers through a ring for future actions (Fig. 13 (a));
- Interactions with a nearly two-dimensional, very thin object (Fig. 13 (b)).

As depicted in Fig. 13 (a), effectively controlling multiple fingers of the hand to pass through the ring of a wristwatch for secure attachment to the palm poses a significant challenge. Presently, our method struggles to provide satisfactory solutions for such cases, possibly due to morphological disparities between the human hand and the robot hand. These differences make it difficult to replicate human-like actions with the robot hand. Additionally, we encounter difficulties achieving desirable outcomes when interacting with extremely thin objects, especially when one dimension of the object scales down to near-zero, as illustrated in Fig. 13 (b). Such challenging object shapes make it challenging to devise an effective lifting strategy.

B.5 User Study

We conduct a supplementary user study to complement the quantitative and qualitative evaluations presented in the main paper, website, and supplementary video, aiming to comprehensively assess and compare the quality of our transferred manipulations with those of the baseline method, DGrasp-Tracking. Our user study is hosted on a website, where the results of our method and DGrasp-Tracking on 10 sequences are presented in a randomly permuted order. Ten participants, regardless of their familiarity with the task or expertise in computer science, are asked to rate each clip on a scale from 1 to 5 to indicate their preferences. Specifically, “1” indicates a significant difference between the transferred motion and the reference motion, “3” represents the manipulation task is completed to some extent but the hand motion deviates obviously from the reference motion, “5” indicates a delicately controlled motion with a good task completeness and human-likeness. Intermediate values of "2" and "4" represent in-between assessments.

For each clip, we calculate the average score achieved by our method and DGrasp-Tracking. The average and median scores across all clips are summarized in Table 2. The results show the significant superiority of our method over the baseline method.

C Experimental Details

C.1 Datasets

Evaluation data comes from three datasets, namely GRAB [18], containing single-hand interactions with daily objects, TACO [12], containing humans manipulating tools, and ARCTIC [7] with bimanual manipulations. We’ll publicly release the dataset for future research.

Table 2: User study.

	Ours	DGrasp-Tracking
Average Score	**4.00**	2.06
Median Score	**3.95**	2.10

Table 3: Default *parameter settings* of the **quasi-physical simulator curriculum**.

Object	`box`	`capsulemachine`	`espressomachine`	`ketchup`	`laptop`	`microwave`	`mixer`	`phone`	`scissors`	`waffleiron`
Subject ID	1	5	6	7	4	1	5	7	4	2

GRAB [18]. We randomly randomly sample a manipulation trajectory for each object. If its manipulation is extremely simple, we additionally sample one trajectory for it. The object is not considered if its corresponding manipulation is bimanual such as `binoculars`, involves other body parts such as `bowl`, or with detailed part movements such as the `game controller`. Finally, manipulations with the following objects are included in our dataset: `mouse`, `flashlight`, `stapler`, `hammer`, `torus`, `stanfordbunny`, `pyramid`, `cylinder`, `airplane`, `train`, `mouse` (resampled), `cube`, `watch`, `waterbottle`, `phone`, `sphere`, `mug`, `alarmclock`, `knife`, `fryingpan`, `cup`, `duck`, `elephant`, `lightbulb`, `scissors`, `toothbrush`, `toothpaste`. For each sequence, we take the first approach-action clip with the length of 60 frames. The number of manipulation sequences from GRAB is 27.

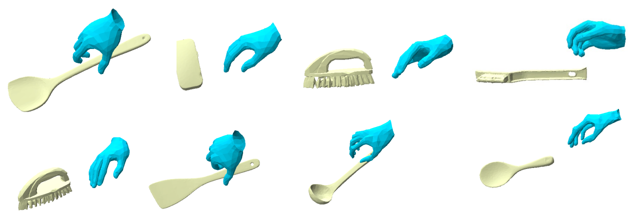


Fig. 14: Snapshots from the TACO dataset.

TACO [12]. For TACO, we acquire data by contacting authors. We randomly select one sequence for each right-hand tool object, a few snapshots are presented in Fig. 14. Sequences with very low quality like erroneous object motions are excluded. For each trajectory, we take the first approach-action clip with the maximal length set to 150 frames. 14 trajectories in total are selected finally.
ARCTIC [7]. For ARCTIC, we randomly select one sequence for each object from its available manipulation trajectories, resulting in 10 sequences in total. For each trajectory, we take the first approach-action clip with the maximal

length set to 150 frames. The object names and the corresponding subject indexes are summarized in Table 3. Please note that subject `s08` and `s09` only have "`use`" actions. Besides, some "`grab`" sequences are missing in a specific subject's manipulation sequences. For instance, both `s01` and `s06` do not have "`grab`" manipulations with `box`.

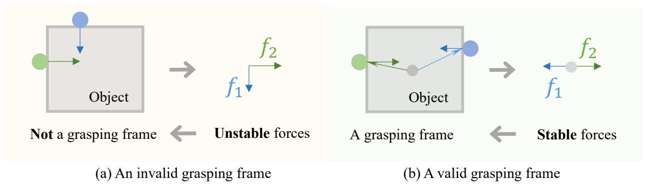


Fig. 15: Grasping frame. We leverage a simple strategy to find the first grasping frame from the sequence. A valid grasping frame should have at least two contact points. The contact force directions should be able to stabilize the object, *i.e.,* there exists a solution for their magnitudes so that zero force and zero torque are applied to the object.

C.2 Baseline

DGrasp-Base [5]. We use the official code provided by authors[2]. We adapt the codebase to two simulated environments used in our evaluation, Bullet [6] and Isaac Gym [13]. Using DGrasp's method to complete the tracking task requires us to define reference grasping frames. We leverage a heuristic method and take the first grasp frame as the reference frame, illustrated in Fig. 15. Specifically, the first grasp is the first frame in the sequence satisfying the following conditions: 1) at least two contacts are detected between the hand and the object, 2) all contact force directions can form a force closure, that is there exists a solution for their magnitudes so that the object is stable under such contact forces. Having defined the reference grasping frame, we train the manipulation policy using the original DGrasp's method. Initially, only the grasping policy is activated. The grasping module guides the hand towards the object to find a stable grasp according to the defined reference frame. After that, the grasping policy and the control policy cooperate to move the object to the final 6D pose. Our method can find a successful policy on DGrasp's "021_bleach_dexycb" example in two simulated environments using the dynamic MANO hand [5].

DGrasp-Tracking (improved from DGrasp [5]). We set a series of reference frames from the sequence, where every two reference frames are separated by 10 frames. We use the grasping policy to guide the hand toward each reference frame.

[2] DGrasp's GitHub Repository.

DGrasp-Tracking (w/ curric.). We gradually train DGrasp-Tracking in each of the simulators from the quasi-physical simulators, finally in the tested simulator. The curriculum setting is the same as that listed in Table 4.

ControlVAE [21]. We adapt the official release[3] to the manipulation scenario. The world model approximates state transitions. It takes the current state, composed of the articulated dexterous robot hand joint state (including the first 6-DoF global rotations and translations), the object state, including the 4-dim object orientation represented as a quaternion, and the 3-DoF object translation, and control signals, including the velocity and position controls for each hand joint, as input. It outputs the predicted delta hand joint states and the predicted object delta rotations (3-DoF) as well as the delta translations (3-DoF). Following ControlVAE [21], the world model is an MLP. We increase the network depth, resulting in an MLP with 9 layers in total. The first hidden dimension is 256, followed by 6 layers with the hidden dimension of 512, 1 layer with the hidden dimension of 256, and the output layer. `ReLU` is used as the activation function between each hidden layer. The policy network takes the current state, including the hand joint state, object orientation as well as object rotation, and the target state, including the target hand joint states, target object orientation as well as the target object rotation as input. It predicts control signals for the articulated hand, including the position and velocity controls for each hand joint. The policy network is an MLP. The number of layers and the hidden dimension settings are the same as the world model. Length of the replay buffer is set to 1024. For Bullet, the batch size is set to 1. At each training loop, the world model is trained for 256 steps, followed by training the policy network for 256 steps. For Isaac Gym, the batch size is set to 128. At each training loop, the world model is trained for 8 steps, followed by training the policy network for 8 steps. Rollout lengths for the world model and for the policy are 24 and 19 respectively. The number of the maximum training iterations is set to 30000.

MPC (w/ base sim.). The base simulator is the final analytical part of the quasi-physical simulator of the physics curriculum. Articulated rigid constraints are imposed. The spring-damper contact model is tuned to the stiffest level. Please refer to Table 4 for the setting of this simulator.

MPC (w/ base sim. w/ soften). Based on the base simulator, we introduce the soften strategy present in Bundled Gradients [17]. Penalty-based contacts are smoothed by sampling contact spring coefficients, as stated in Section IV.B [17]. The sampling range for each coefficient is defined as the [-10%, +10%] interval of the original value.

C.3 Experimental Settings

The quasi-physical simulator curriculum. By default, the curriculum is composed of ten parameterized quasi-physical simulators. We summarize their parameter settings in Table 4. The contact distance threshold d^c, contact spring

[3] ControlVAE's GitHub Repository.

Table 4: Default *parameter settings* of the **quasi-physical simulator curriculum**.

Simulator ID	1	2	3	4	5	6	7	8	9	10
Point Set Parameter α	0.1	0	0	0	0	0	0	0	0	0
Contact Distance Threshold d^c	0.1	0.1	0.05	0.03	0.025	0.02	0.015	0.01	0.0	0.0
Contact Spring Stiffness k^n	4×10^4	4×10^4	8×10^4	1×10^5	2×10^5	3×10^5	3.5×10^5	4×10^5	4×10^6	4×10^6
Friction Spring Stiffness k^f	1×10^5	1×10^5	2×10^5	4×10^5	5×10^5	6×10^5	8×10^5	1×10^6	1×10^7	1×10^7
Contact Damping Coefficient k^d	1×10^3	1×10^3	1×10^3	1×10^3	1×10^3	1×10^3	1×10^3	1×10^3	1×10^3	1×10^3
w/ Residual Physics?	No	No	No	No	No	No	No	No	No	Yes

Table 5: *Curriculum parameter settings* used in the ablated version (Ours w/ Curriculum II).

Simulator ID	1	2	3	4	5	6	7
Point Set Parameter α	0.1	0	0	0	0	0	0
Contact Distance Threshold d^c	0.1	0.1	0.05	0.02	0.01	0.0	0.0
Contact Spring Stiffness k^n	4×10^4	4×10^4	8×10^4	3×10^5	4×10^5	4×10^6	4×10^6
Friction Spring Stiffness k^f	1×10^5	1×10^5	2×10^5	6×10^5	1×10^6	1×10^7	1×10^7
Contact Damping Coefficient k^d	1×10^3	1×10^3	1×10^3	1×10^3	1×10^3	1×10^3	1×10^3
w/ Residual Physics?	No	No	No	No	No	No	Yes

stiffness k^n, friction spring stiffness k^f, and contact damping coefficient k^d are set empirically.

For the ablated version ("Ours w/ Curriculum II" in the ablation study), we remove some stages from the original curriculum. The setting is summarized in Table 5.

Quasi-physical simulators. We use `Python` to implement each component of the simulator and the simulation processes, including the articulated rigid dynamics, the point set dynamics, the spring-damper contact modeling, and the residual physics modules. Semi-implicit time-stepping is leveraged. Time stepping is set to 5×10^{-4} with 100 substeps per frame. In this way, we can easily introduce neural network components into the simulator. Besides, one can easily integrate it into a deep learning framework for further applications. Moreover, we can calculate gradients automatically taking advantage of the auto-grading feature of the framework. The overall efficiency, though has a large improvement space, is acceptable in our task.

Converting meshes to SDFs. We use Mesh2SDF [19] in this process.

Parameters set $\mathcal{S}$. The parameter set $\mathcal{S}$ includes object properties, *i.e.,* object mass and object inertia, and some unknown system parameters, *i.e.,* linear velocity sampling coefficient and angular velocity damping coefficient. For the friction coefficient, we set it to a fixed value, *i.e.,* $\mu = 10$. The value is set under the consideration of the important role friction forces play in the manipulation task.

Controlling the hand in Bullet and Isaac Gym. In our quasi-physical simulator, the hand is controlled via joint forces and root linear and angular velocities. In Bullet and Isaac Gym, people commonly use PD controls, which are also recommended officially [6]. Therefore, to convert controls in joint forces and root velocities to PD controls in the them, we additionally add a control transformation module.

For each timestep $1 \leq n \leq N - 1$, it takes root positions at the timestep n and $n + 1$, joint states, velocities, joint forces, and the object state at step n and outputs the residual position and velocity controls at step n. The predicted residual PD controls added to the root positions, root velocities (calculated via finite differences), joint states, and velocities are treated as PD controls in the target simulator. The control transformation module is composed of a hand point feature extraction layer, an object feature extraction layer, and a prediction layer. The current hand and object geometry is firstly encoded in latent features. Subsequently, the original joint control related information and the encoded latent features are fed into an MLP for residual position and velocity control prediction. The feature extraction layer is a 3-layer MLP with hidden dimensions [128, 128, 128] and `ReLU` as the activation layer. After per-point feature extraction, a `maxpool` function operates on point features to extract global features for the hand and the object. Then the global features of the hand and the object are concatenated together and passed through a two-layer MLP with hidden dimension 128 and the output dimension 128 as well. The output feature is then concatenated with the object control related information and passed through an MLP for the residual control prediction. The prediction network is a 3-layer MLP with hidden dimension [128, 64]. The control transformation module is optimized together with the residual physics module introduced in the parameterized quasi-physical simulator.

World model-style training. Rollout lengths for both the trajectory optimization and the model training are set to 19. In each iterative training iteration, the trajectory is optimized for 256 steps. The residual physics module is optimized for 256 steps. The replay buffer length is 1024.

Evaluation process. Our method is a multi-stage optimization-based strategy. The overall optimization process can be roughly divided into three stages, as illustrated in the following:

- **Transferring via point dynamics.** This stage involves three processes:
  - Optimize a dynamics MANO [5] trajectory that can track the input kinematics-only trajectory;
  - Optimize the control trajectory for the point set of the MANO hand that can track the hand trajectory and the object trajectory;
  - Optimize a kinematics-only trajectory for the simulated robot hand so that it can track the kinematic hand trajectory via sparse correspondences;

  - Optimize the control trajectory for the point set of the simulated robot hand so that it can track the trajectory of the MANO's point set.
- **Optimizing through a contact curriculum.** In this stage, the control trajectory of the simulated robot hand is optimized in each simulator from the curriculum. The objective is to track the hand trajectory and the object trajectory.
- **Transferring to a realistic simulated environment.** In this stage, the quasi-physical simulator and the control trajectory for the simulated robot hand are iteratively optimized. By default, the number of iterations is set to 30,000.

In each optimization iteration, excluding the kinematics trajectory-only optimization, the parameter set $\mathcal{S}$ and the control trajectory are optimized alternately. If we cannot inherit a control trajectory from the previous stage, we first optimize the it with the parameters $\mathcal{S}$ either inherited from previous stages or set to default values. After that, the parameters $\int$ are further refined with controls fixed. Subsequently, we continue to optimize controls based on the identified parameters. If the control trajectory can be inherited from previous stages at the beginning of the iteration, the parameters $\mathcal{S}$ are identified with controls fixed. Then we further refine controls with parameters fixed. Typically, the number of optimization steps for the parameters is 1000, while the number is 100 for the control trajectory. Both hand controls and parameters are optimized via gradient descent. Learning rate is set to 5×10^{-4} for both control optimization and parameters identification. We use `Adam` optimizer. No learning rate scheduler is used. In the third stage, we follow the training framework in ControlVAE [21]. The optimizer is `RAdam`, with the learning rate 10^{-4} for the quasi-physical simulator and 10^{-4} for control trajectory optimization.

C.4 Running Time and Complexity

Complexity. The time complexity is related to the number of frames in the manipulation sequence and the number of optimization passes. Denote the number of frames as N and the number of total optimization passes as K, the time complexity is $\mathcal{O}(KN)$.

Running time. Taking a sequence with 60 frames as an example, the first stage (see **evaluation process** stated in the previous section) costs about 7 hours in total. Using the default curriculum setting (Table 4), the second stage would cost about 22 hours. Early termination logic in each optimization iteration will shorten the time. Therefore, the actual time is per-sequence dependent. Taking transferring to the Bullet simulator as an example, the third stage takes about 20 hours to complete. Reducing the number of simulators in the curriculum or using a smaller number of iterations in the third stage can improve the time efficiency.

D Potential Negative Societal Impact

Our approach has the potential to expedite the advancement of robotic dexterous manipulation skills. However, in the future, the emergence of highly developed robots proficient in performing various tasks may lead to the replacement of certain human labor, thus potentially impacting society.

References

1. Arunachalam, S.P., Silwal, S., Evans, B., Pinto, L.: Dexterous imitation made easy: A learning-based framework for efficient dexterous manipulation. In: 2023 ieee international conference on robotics and automation (icra). pp. 5954–5961. IEEE (2023) 13
2. Bahl, S., Mendonca, R., Chen, L., Jain, U., Pathak, D.: Affordances from human videos as a versatile representation for robotics. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13778–13790 (2023) 12, 13
3. Bharadhwaj, H., Gupta, A., Kumar, V., Tulsiani, S.: Towards generalizable zero-shot manipulation via translating human interaction plans. arXiv preprint arXiv:2312.00775 (2023) 12
4. Chen, G., Cui, T., Zhou, T., Peng, Z., Hu, M., Wang, M., Yang, Y., Yue, Y.: Human demonstrations are generalizable knowledge for robots. arXiv preprint arXiv:2312.02419 (2023) 12
5. Christen, S., Kocabas, M., Aksan, E., Hwangbo, J., Song, J., Hilliges, O.: D-grasp: Physically plausible dynamic grasp synthesis for hand-object interactions. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 20577–20586 (2022) 5, 17, 20
6. Coumans, E., Bai, Y.: Pybullet, a python module for physics simulation for games, robotics and machine learning (2016) 17, 20
7. Fan, Z., Taheri, O., Tzionas, D., Kocabas, M., Kaufmann, M., Black, M.J., Hilliges, O.: ARCTIC: A dataset for dexterous bimanual hand-object manipulation. In: Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2023) 15, 16
8. Featherstone, R.: Rigid body dynamics algorithms (2007), https://api.semanticscholar.org/CorpusID:58437819 4
9. Garcia, C.E., Prett, D.M., Morari, M.: Model predictive control: Theory and practice—a survey. Automatica **25**(3), 335–348 (1989) 6
10. Guo, D.: Learning multi-step manipulation tasks from a single human demonstration. arXiv preprint arXiv:2312.15346 (2023) 12
11. Liu, C.K., Jain, S.: A quick tutorial on multibody dynamics. Online tutorial, June p. 7 (2012) 4
12. Liu, Y., Yang, H., Si, X., Liu, L., Li, Z., Zhang, Y., Liu, Y., Yi, L.: Taco: Benchmarking generalizable bimanual tool-action-object understanding. arXiv preprint arXiv:2401.08399 (2024) 15, 16
13. Makoviychuk, V., Wawrzyniak, L., Guo, Y., Lu, M., Storey, K., Macklin, M., Hoeller, D., Rudin, N., Allshire, A., Handa, A., et al.: Isaac gym: High performance gpu-based physics simulation for robot learning. arXiv preprint arXiv:2108.10470 (2021) 17

14. Qin, Y., Huang, B., Yin, Z.H., Su, H., Wang, X.: Dexpoint: Generalizable point cloud reinforcement learning for sim-to-real dexterous manipulation. In: Conference on Robot Learning. pp. 594–605. PMLR (2023) 12, 13
15. Qin, Y., Wu, Y.H., Liu, S., Jiang, H., Yang, R., Fu, Y., Wang, X.: Dexmv: Imitation learning for dexterous manipulation from human videos. In: European Conference on Computer Vision. pp. 570–587. Springer (2022) 12, 13
16. Shaw, K., Bahl, S., Pathak, D.: Videodex: Learning dexterity from internet videos. In: Conference on Robot Learning. pp. 654–665. PMLR (2023) 12
17. Suh, H.J.T., Pang, T., Tedrake, R.: Bundled gradients through contact via randomized smoothing. IEEE Robotics and Automation Letters **7**(2), 4000–4007 (2022) 18
18. Taheri, O., Ghorbani, N., Black, M.J., Tzionas, D.: Grab: A dataset of whole-body human grasping of objects. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IV 16. pp. 581–600. Springer (2020) 15, 16
19. Wang, P.S.: Mesh2sdf: Converts an input mesh to a signed distance field (2022), https://github.com/wang-ps/mesh2sdf 19
20. Xu, J., Chen, T., Zlokapa, L., Foshey, M., Matusik, W., Sueda, S., Agrawal, P.: An end-to-end differentiable framework for contact-aware robot design. arXiv preprint arXiv:2107.07501 (2021) 11
21. Yao, H., Song, Z., Chen, B., Liu, L.: Controlvae: Model-based learning of generative controllers for physics-based characters. ACM Transactions on Graphics (TOG) **41**(6), 1–16 (2022) 18, 21
22. Zhang, H., Christen, S., Fan, Z., Zheng, L., Hwangbo, J., Song, J., Hilliges, O.: Artigrasp: Physically plausible synthesis of bi-manual dexterous grasping and articulation. arXiv preprint arXiv:2309.03891 (2023) 12